

Plenge
64
Das
SUPER
GRAFIK
Buch zum Commodore 64
komplett mit Diskette
DATA BECKER

Plenge

Das SUPER GRAFIK Buch zum Commodore 64

Ein DATA BECKER Buch

Komplett mit Diskette

ISBN 3-89011-213-7

2. leicht unveränderte Auflage.

Vorwort

Diese völlig neue Auflage des alten Graphikbuches zum COMMODORE 64 bietet mit einem revolutionär neuen Konzept wirklich Neues. Zusammen mit dem dicken und schweren Supergraphik-Buch erhält der Graphik-Interessierte ein professionelles Programm, das sonst nur einzeln und für sich erhältlich war: Eine neue, verbesserte Version der bekannten und erfolgreichen Supergraphik.

Zudem wird das komplette Source-Listing dieser Supergraphik abgedruckt und die einzelnen Routinen somit jedem zugänglich vorgestellt. Sie erhalten also nicht nur detaillierte Informationen zur Realisierung eigener Graphiken und zur Einbindung von Graphik-Routinen in eigene Programme, Sie haben sogar die Möglichkeit, eine komfortable BASIC-Erweiterung direkt einzusetzen.

Die neue Version der Supergraphik bietet neben den bekannten Befehlen und Features (16 Sprites voll bewegbar, Text-Graphik-Mischanzeige, Graphik-Hardcopies für eine Vielzahl von Drukkern etc.) einen Befehl zum Ausfüllen beliebiger Flächen sowie einen ganzen Befehlskomplex zur Erzeugung von Bildschirmfenstern auf Ihrem Rechner.

Grafik ist nicht nur eine der Hauptstärken des COMMODORE 64; COMMODORE hat diese Stärke auch sehr sorgfältig versteckt. Der BASIC-Anfänger kennt Grafik nur bewundernd von den vielen fertigen Programmen und Aktionspielen, die natürlich über entsprechende Routinen - meist in Maschinensprache - die grafischen Fähigkeiten des COMMODORE 64 voll ausnutzen. Dieses Buch soll nun jedem COMMODORE 64-Anwender die Möglichkeit geben, auch in eigenen Programmen die Vielzahl graphischer Möglichkeiten seines Computers zu nutzen.

Schauen Sie sich doch einmal das Inhaltsverzeichnis an, um sich von der Vielseitigkeit des Supergraphik-Buches zu überzeugen.

Bevor wir uns ins Getümmel werfen, möchte ich an dieser Stelle noch einen Dank an Klaus Löffelmann richten, der an verschiedenen Stellen durch Arbeit unterstützte.

Düsseldorf, im November 1986 — Axel Plenge

Inhaltsverzeichnis

1. Einleitung

"64 K RAM, 8 unabhängige Sprites, frei auf dem Bildschirm bewegbar, hochauflösende Graphik mit 320x200 Punkten, Multicolorgraphik (160x200 Punkte), 16 Farben, 40 Zeichen, 25 Zeilen, veränderbarer Zeichensatz, sensationelle Interruptmöglichkeiten, ..., ein Computer, den jeder kennenlernen, nein, besitzen muß!"

So oder ähnlich wird - wohl auch zu Recht - unser guter alter Commodore 64 angepriesen. Kaum jemand, der sich auch nur ein wenig mit Computern auskennt, wird an solchen Anzeigen vorbeischauen, denn Ihr 64er kann wirklich viel.

Doch bald nach dem Kauf und dem hoffnungsvollen Studium der Betriebsanleitung aber bekommt man ein merkwürdiges Gefühl in der Magengrube: Kein Wort von der hochauflösenden Graphik, geschweige denn von solchen Vorzügen wie Zeichensatzveränderung oder Interruptfähigkeiten. Selbst das für die Maßstäbe des Handbuches relativ ausführliche Kapitel über die Sprites und ihre Programmierung verschleiert die wahren Möglichkeiten des Gerätes.

Doch wir sollten nicht zu streng mit dieser Kurzanleitung zu Gerichte ziehen. Um auch nur annähernd alle Vorzüge unseres Rechners zu beschreiben und gar ihre Anwendung zu demonstrieren, bedarf es einer etwas umfangreicheren Lehrbuchkonzeption.

An dieser Stelle greift nun das vorliegende Graphikbuch zum 64er an. Dieses Buch soll Ihnen umfassende Kenntnisse über das graphische Innenleben eines Computers vermitteln, der nicht umsonst zum Computer des Jahres 1983 gewählt wurde.

Das vorliegende Graphikbuch unterliegt im Vergleich zu anderen einer völlig neuen Konzeption. Das neue Graphikbuch wird direkt mit einer beiliegenden Diskette geliefert, die alle wesentlichen hier abgedruckten Programme enthält. Zusätzlich, und das ist das ganz neue und geradezu sensationelle an diesem Werk, wird die Graphik-BASIC-Erweiterung SUPERGRAPHIK 64 auf dieser Diskette mitgeliefert. Aber nicht nur das; zusätzlich dazu wird das gesamte Source-Listing dieser Supergraphik vollständig dokumentiert in diesem Buch abgedruckt! Damit wird Graphik für jeden zugänglich und verständlich.

Dieses Buch ist also nicht nur für die Commodore-Besitzer interessant. Die abgedruckten Routinen der Supergraphik stellen eine Graphik-Programmbibliothek für alle Programmierer dar, egal für welchen Computer.

In jedem Teil des Buches wurde darauf geachtet, daß alle Zusammenhänge möglichst einfach und verständlich dargestellt wurden. Die Lernschritte sind gegliedert und gut zu überblicken.

Damit soll das Buch gerade dem Einsteiger eine gute Hilfe sein. Die Fortgeschrittenen unter Ihnen kommen jedoch (wie Sie gesehen haben und sehen werden) ganz bestimmt nicht zu kurz!

Das Buch unterteilt sich im wesentlichen in fünf große Sinnabschnitte: Im ersten dieser Abschnitte (das 2. Kapitel) erhalten Sie anhand der verschiedenen Befehle der Supergraphik eine Einführung in die verschiedensten Graphikbereiche. Alle Befehle werden hier vorgestellt, eine ganze Menge interessanter und nützlicher Programme demonstrieren nicht nur die Anwendung dieser Befehle, sie vermitteln auch Tips und Tricks zur allgemeinen Graphikanwendung.

Das dritte Kapitel wendet sich an diejenigen, die Graphik für fortgeschrittenere Themen einsetzen wollen (Statistik, 3dimensionale Graphik, Spiele etc.). Dabei werden sich die meisten dieser Programme auf die Befehle der Supergraphik stützen. Kurz, hier entdecken Sie den tatsächlichen Nutzen des 64ers; auf diesen Abschnitt können Sie verweisen, wenn Sie jemand fragt: "Ein Computer? Was soll ich damit?"

Damit Sie aber Graphik nicht nur von der Supergraphik-Ebene, also aus einem erweiterten BASIC heraus anwenden können, widmet sich das vierte Kapitel den graphischen Möglichkeiten Ihres Rechners überhaupt. Hier finden Sie alles, was Sie für Graphik auf dem 64er benötigen. Dieses Kapitel wird später wahrscheinlich (neben dem 2. Kapitel) das von Ihnen meist "beblätterte" sein, da es sich hervorragend als Nachschlagewerk eignet.

Hier erfahren Sie alles (wirklich alles), was es über die Bildschirmsteuerung durch den VIC (Videocontroller), dem Organisator des gesamten Bildgeschehens, zu erfahren gibt. Hier sehen Sie, welche Bedeutung beispielsweise den vielen Registern dieses "Managers" zukommt, was Sprites sind und wie sie organisiert sind, was man tun muß, um Graphik einzuschalten, zu erstellen und zu bedienen. Ihnen wird endlich klar, was es mit der bisher ziemlich undurchsichtig erscheinenden Graphik-, Zeichensatz- oder Bildschirmspeicher-

Verschiebung auf sich hat, allgemein wie der gesamte Speicheraufbau des 64ers beschaffen ist und welche Möglichkeiten sich ergeben...

Im fünften Kapitel schließlich erhalten Sie einen Überblick über die programmiertechnische Seite der Graphik. Wie werden Punkte, Linien und Kreise am schnellsten, am komfortabelsten, am genauesten gezeichnet? Wie programmiere ich Sprites und Zeichensatz? Hierzu werden Ihnen zwei sehr komfortable Editorprogramme zur Verfügung gestellt, mit denen Sie ohne viele Mühe bequem und handlich diese Dinge erledigen können. Sprites werden auf dem Bildschirm bewegt und auf Kollisionen etc. überprüft. Weitere Kapitel führen Sie in das Laden, Speichern und die Erstellung von Hardcopys der Graphik ein und erläutern Ihnen die Interrupttechniken.

Sie erhalten einen Überblick über die Algorithmen der Graphikprogrammierung mit jeweiligen Verweisen auf das Source-Listing der Supergraphik, das Sie schließlich und letzten Endes in Kapitel 6 finden.

Damit Sie leichter Ihre eigenen Graphikroutinen auch ohne die Supergraphik in Ihre Programme einbauen können, wird zusätzlich zur Supergraphik noch ein kleines, kompaktes Graphikpaket mit den wichtigsten Befehlen abgedruckt.

Wir hoffen, damit jeden 64er-Anwender, Einsteiger wie Fortgeschrittenen, zu einem absoluten "Graphik-Freak" zu machen, einem Fachmann in allen Fragen, der zu Rate gezogen wird, immer dann, wenn andere nicht mehr weiter wissen. Und sollte doch einmal eine Gedächtnislücke auftreten, so schlägt er kurz einmal im Anhang nach und schon weiß er bescheid. Umfangreiche Hinweise auf weiterführende Literatur runden das Buch ab.

2. Supergraphik für Einsteiger

Die ersten Graphik-Gehversuche

In den folgenden Abschnitten werden wir gemeinsam in die Geheimnisse der Computergraphik mithilfe der Befehle der Supergraphik einsteigen. Wir werden anhand kleiner oder größerer Beispielprogramme die verschiedenen Effekte und Techniken kennenlernen, die Graphik an sich ausmachen.

In diesem zweiten Kapitel beschäftigen wir uns vornehmlich mit einfachen Graphikstrukturen und deren Anwendungen, die sich aus der Erklärung der Supergraphik-Befehle ergeben. Dabei werden wir - neben den vielen anderen Demonstrationsprogrammen - ein größeres Projekt in Angriff nehmen, das sich durch das gesamte zweite Kapitel ziehen wird. Wir werden im Laufe der Zeit ein teilweise Menü-gesteuertes schönes CAD-Zeichenprogramm entwickeln, mit dem Sie am Ende dieses Kapitels Ihre eigenen Graphiken direkt am Bildschirm erzeugen können. Dabei werden Sie in der Lage sein, dieses Programm nachträglich Ihren eigenen Bedürfnissen selbst anzupassen (z.B. durch Steuerung mittels Lightpen, Joystick oder Graphiktablett). Diese Graphiken können Sie dann auf Ihrem Drucker ausdrucken oder auf Diskette speichern, um es z.B. später oder in anderen Graphikprogrammen weiter zu bearbeiten.

2.1 Supergraphik 64 - Eine Vorstellung

Nehmen Sie doch einmal die Diskette aus der Buchhülle. Da liegt sie also nun vor Ihnen, die BASIC-Befehlserweiterung Supergraphik 64. In ihr werden Sie Dinge finden, die bisher in keinem einzigen Programm verwirklicht wurden, zumal als für jeden "BASIC-Sprechenden" gut zugängliche BASIC-Befehlserweiterung. Lassen Sie mich Ihnen kurz die allerwesentlichsten Merkmale der neuen Supergraphik vor Augen führen, um Ihnen einen schnellen Überblick über die Leistungsfähigkeit dieses Programmes zu geben. Konnten in früheren Versionen noch Angaben über die Anzahl der Befehle und Befehlskombinationen gemacht werden (damals ca. 183), so ist es nun ein recht hoffnungsloses Unterfangen angesichts der Vielzahl der Variationsmöglichkeiten allein eines Befehls:

- zwei unabhängige hochauflösende (320x200) oder Multicolor- (160x200) Graphikseiten.

- eine Standard-Low-Graphik-Seite (80x50 Punkte) gemeinsam im Textmodus

- Verdecktes Zeichnen (z.B. Text sichtbar, Graphikseite 2 wird erstellt etc.)

- Bildschirmfensterbehandlung

- 16 (!) unabhängige und unterschiedliche Sprites gleichzeitig auf dem Bildschirm und in allen Parametern veränderlich durch Bildschirmfenstertechnik (Windowing).

- Alle 16 Sprites definiert durch einen Befehl!

- Positionieren und Bewegen(!) von 8 bzw. 16 Sprites gleichzeitig und unabhängig voneinander, während das übrige Programm weiterläuft, durch bedingungslosen Einsatz der Interrupttechnik.

- Ständige Sprite-Kollisionsüberprüfung, Joystickunterstützung, Paddleabfrage (A/D-Wandler), dadurch Möglichkeit des Einsatzes z.B. des Koala Graphiktabletts z.B. zum Zeichnen.

- Automatische Unterbrechung des BASIC-Programms bei Kollisionen (Interrupt), Sprung in Unterbrechungsroutine, dann Weiterführung des Hauptprogrammes.

- Für jede Anwendung kombinierbare Befehle und Befehlskombinationen (alle Befehle anwendbar auf alle Graphikarten):

1.) Für alle Zeichenbefehle wählbare Zwischenmodi:
Zeichnen,
Löschen,
Invertieren,
Punktieren,
Pinselwahl,
Graphikcursor bewegen,
Zeichnen mit/ohne Farbsetzung,
Punkte zählen.

2.) Durch einfache Befehle zu steuernde
Graphikfiguren:
Punkt,
Linie,
Linienschar,
Linie vom Graphikcursor,
Kreise,
Kreisbögen,
Ellipsen,
Ellipsenbögen,
Vielecke (geschlossen und offen),
selbstdefinierbare Figuren,
rotieren und vergrößern dieser Figuren,
Rahmen,
Felder,
Text in Graphik.
Beliebige Flächen ausfüllen

3.) Weitere Graphikbefehle:
Graphikseiten- und Moduswechsel,
Graphik und Bildschirmfenster maskiert löschen,
Graphik und Bildschirmfenster maskiert invertieren,
Kopieren einer Graphikseite in die andere,
ODER-, UND- und EXKLUSIV-ODER-Verknüpfung der beiden Graphikseiten
Kopieren und Verschieben von Graphikfenstern
ODER-, UND- und EXKLUSIV-ODER-Vernküpfung von Bildschirmfenstern
Originalgetreues Kopieren eines TEXT-Bildschirms in eine Graphikseite (!), dadurch Hardcopy auch des Text-Bildschirms,
Scrolling von Text- und Graphik
Wählen der Rahmen-, Hintergrund-, Zeichen- oder Punktfarbe.

- Gleichzeitige Anzeige von Text und Graphik durch Bildschirmfensterdefinition!

- Speichern und Laden von Graphik, Farbe und Text (!) auf Diskette (Kassette) (auch verdeckt) in selbst wählbarem Format, dadurch Möglichkeit des Einladens aller "Fremd"-Graphiken und deren weitere Bearbeitung durch Hardcopy etc. (auch z.B. aller Koala-Pad-Bilder)

- Hardcopies für EPSON mit DATA-BECKER-Interface, CBM 1526, CBM MPS 801, Seikosha GP 100 VC, Farb(!)drucker Seikosha GP 700 (farbige Hardcopies von Koala-Pad-Bildern etc.) und andere.

- Komfortable Soundprogrammierung mit Verstellung aller möglichen Sound-Parameter (Lautstärke, Klang, Filter, Tonhöhe, Tonlänge) ebenfalls unabhängig vom übrigen Programmablauf.

- Eingebautes Programmier-Set mit:
  RENUMBER (auch GOTO, GOSUB, ON...GOTO, ON...GOSUB, RUN, LIST, THEN-Änderung), MERGE (anhängen eines BASIC-Programmes an ein anderes), DIRECTORY (anzeigen des Disketten Inhaltsverzeichnisses ohne Programmverlust(!), DATASET (setzen des DATA-Zeigers auf eine bestimmte Zeile (auch indirekt), KEY (Programmierung der 8 Funktionstasten mit 16(!) Zeichen pro Taste.

- Cursorpositionierung im Textmodus.

- Erleichtertes Programmieren durch "tolerante" Fehlerakzeptanz (ILLEGAL QUANTITY etc.).

2.2 Starten der Supergraphik

Nach dem Einschalten von Floppy und Computer legen Sie Ihre Diskette ein und tippen:

```
LOAD "SUPERGRAPHIK",8 (return)

RUN (return)
```

und schon wird Ihre Graphikerweiterung gestartet.
Nach einiger Zeit erscheint auf Ihrem Bildschirm ein Menü, in dem Sie Ihrer Supergraphik angeben, welchen Drucker Sie besitzen. Anhand der von Ihnen nun einzugebenden Kennzahl läd Ihre Supergraphik die passende Hardcopyroutine ein, damit Sie auch auf Ihrem Drucker einen Graphikausdruck erzeugen können.

Nachdem Sie die Ihrem Drucker zugeordnete Ziffer eingegeben haben, erscheint dann die Meldung und Sie befinden sich im normalen Programmiermodus, womit Sie sofort anfangen können.

2.3 Kleines Graphik-ABC

Dieses Kapitel können Sie ruhig überschlagen, falls Sie nicht interessiert sind. Trotzdem wird es Ihnen bei dem Verständnis der nachfolgenden Programme in diesem Buch sehr hilfreich sein, sich mit diesem Thema einmal ganz kurz auseinandergesetzt zu haben. Wenn Sie nicht alles verstehen sollten, so ist das nicht weiter schlimm. Weiter hinten im Buch werden diese Dinge sehr ausführlich noch einmal aufgegriffen. Im folgenden sollen Ihnen nämlich einige wichtige Eigenheiten Ihres Rechners in sehr knapper Form dargelegt werden.

Supergraphik 64 bietet Ihnen 3 Graphikarten an, die Sie nach Bedarf auswählen können:

2.3.1 80x50-Blockgraphik (LGR):

Allgemeines:

Diese Graphikart wird zusammen mit dem normalen Text angezeigt und basiert auf der Verwendung folgender 8 Graphiksonderzeichen des CBM 64 jeweils im normalen oder inversen Zustand:

ASCII: 32,161,162,172,187,188,190,191

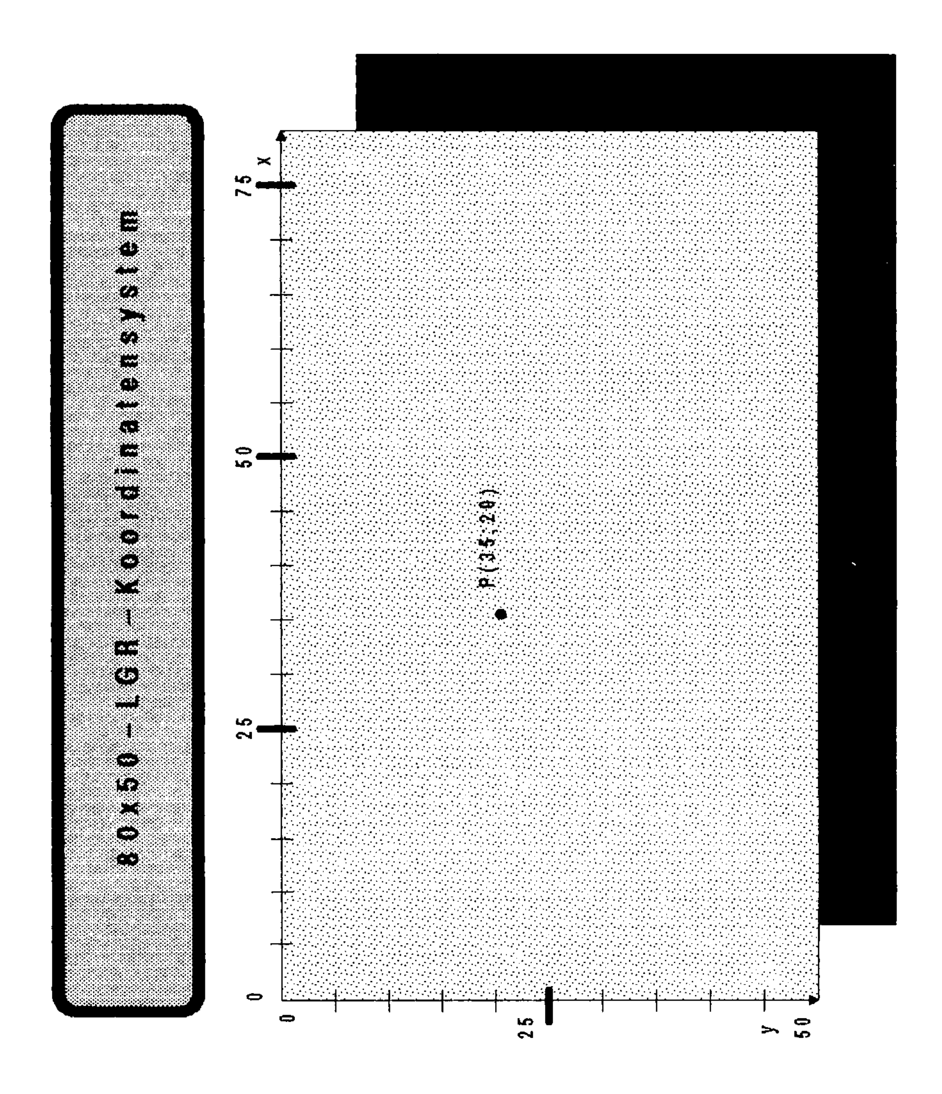
Diese Zeichen können Sie sich mit dem der BASIC-Anweisung

```
PRINT CHR$(32);CHR$(161);CHR$(162);CHR$(172);CHR$(187);CHR$(188);CHR$(19
0);CHR$(191)
```

anschauen.

Die Auflösung beträgt also jeweils 1/4 Zeichen (4x4-HGR-Punkte). Bestehender Text wird nicht überschrieben, so daß Sie ohne weiteres Graphiken und Text miteinander mischen können.

Das zum Setzen eines Punktes notwendige Koordinatensystem sieht dann folgendermaßen aus:


80x50 - LGR - Koordinatensystem
0
25
50
75
x
y
P(35;20)

Auffällig ist, daß sich der Nullpunkt dieses Koordinatensystems oben links in der Ecke des Bildschirms befindet. Diese Einteilung hat sich in der Computerwelt geradezu standardisiert und wird uns noch in vielen Anwendungen und Programmen verfolgen.

Speichernutzung:

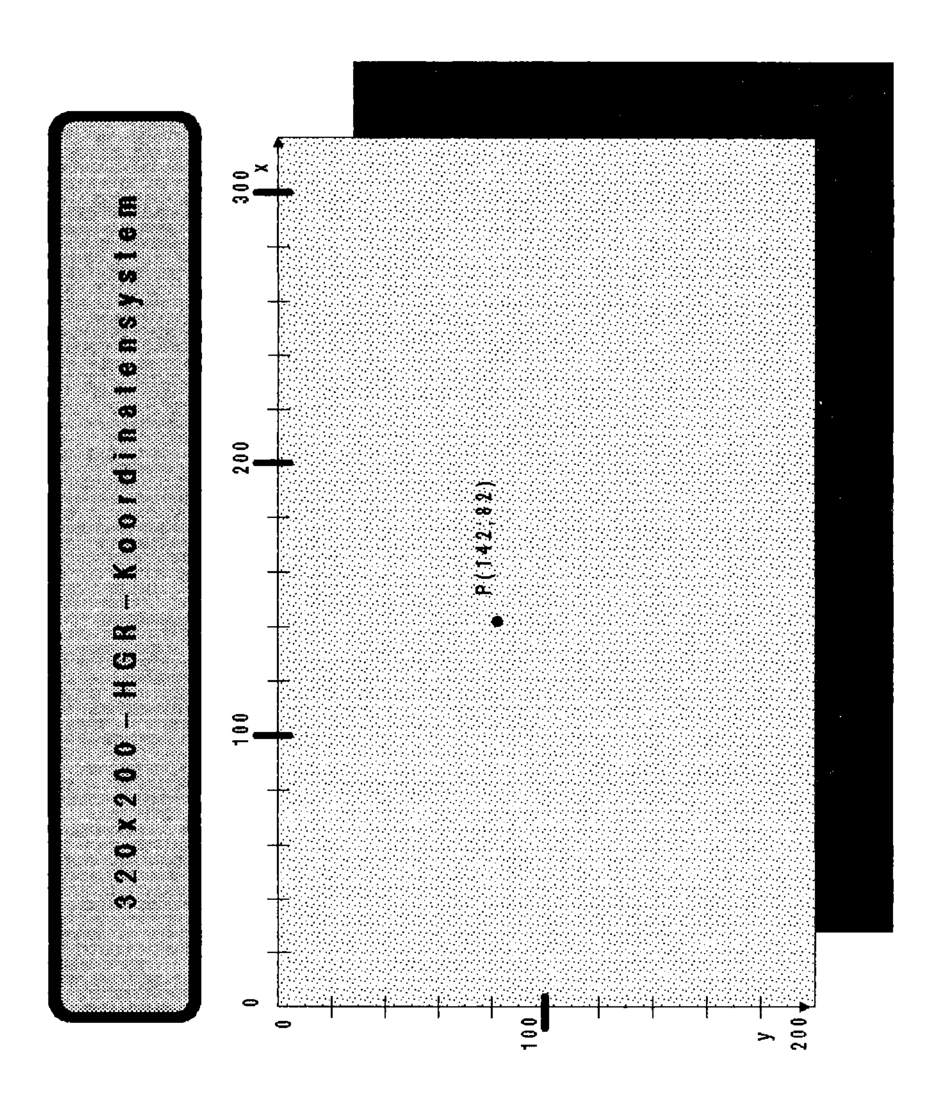
Die Graphikzeichen werden wie der Text im Video-RAM bei den Adressen 1024-2023 ($0400-$07E7) (ca. 1K-Byte) abgelegt. Die Hintergrundfarbe steht im 33. Register des VIC (Video-Interface-Chip) (s. Anhang). Die Zeichenfarbe steht im normalen Farb-RAM bei den Speicherstellen 55296-56295 ($D800-$DBE7).

2.3.2 320x200-Hochauflösende Graphik (HGR):

Allgemeines:

In diesem Graphikmodus ist jeder kleinste Punkt des Bildschirms einzeln ansprechbar, was zu dieser hohen Auflösung führt (ein Buchstabe/Zeichen besteht aus 8x8-Punkten). Ihr CBM 64 benutzt dabei ein Bit jedes Bytes des Graphikspeichers als Punktraster (0=Punkt nicht gesetzt; 1=Punkt gesetzt). Die Farbauflösung ist hier jedoch kleiner: jeweils 8x8 Punkte besitzen die gleiche Farbe (0-15), was zu einer besonderen Farbbehandlung führt (s.u.).

Das zum Setzen eines Punktes notwendige Koordinatensystem sieht dann folgendermaßen aus:


320x200 - HGR - Koordinatensystem
P(142,82)
0
100
200
300
x
y

Speichernutzung:

Es bedarf etwa 8 K-Byte (8000 Bytes), um ein vollständiges Graphikbild zu speichern. Diese 8K werden in der Supergraphik platzsparend positioniert, indem sie für Bildseite 1 (s.u.) ab Speicherstelle 57344 ($E000) und für Bildseite 2 ab Adresse 40960 ($A000) in das unter dem Betriebssystem liegende RAM abgelegt werden, daher also keinen BASIC-Speicherplatz einnehmen.

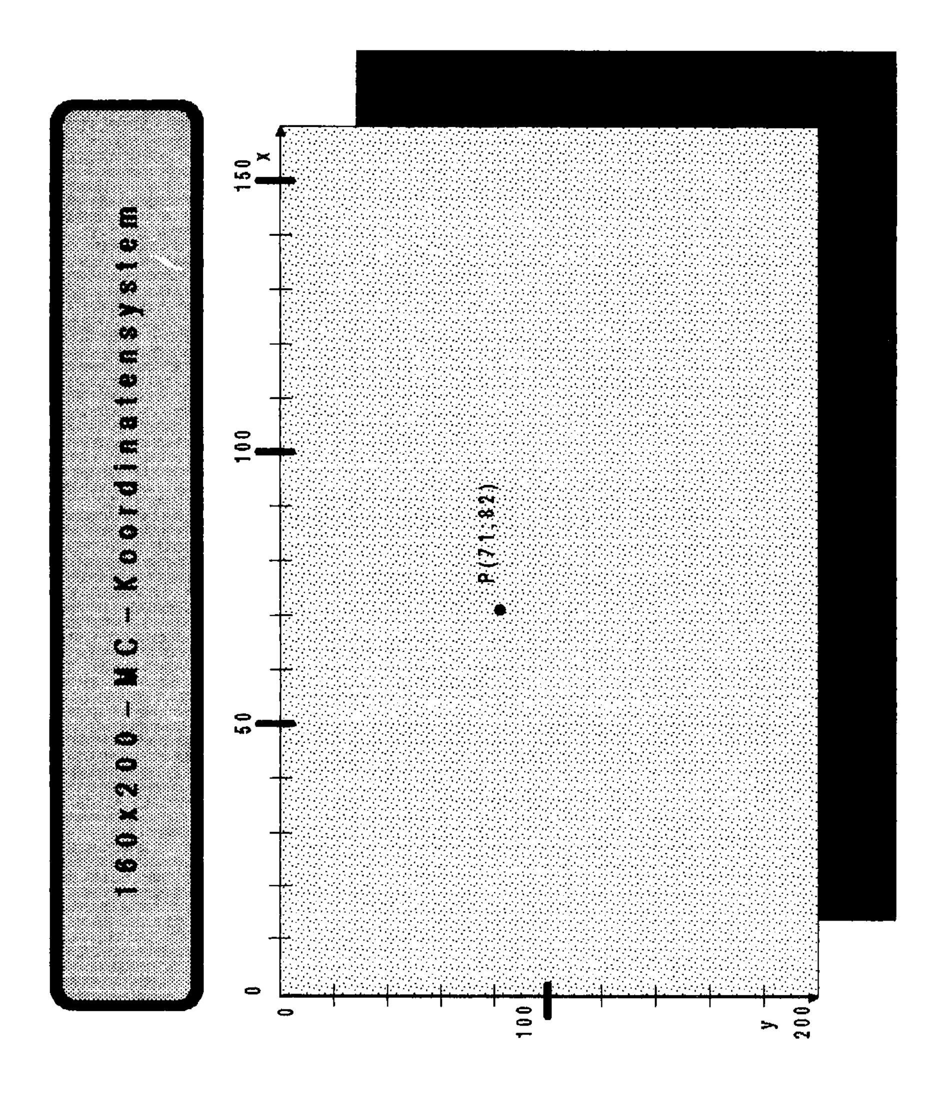
Die Hintergrundfarbe kommt aus den unteren 4 Bits jedes Bytes des Video-RAMs, der nach 49152 ($C000), also ebenfalls aus dem BASIC-Bereich heraus, verschoben wurde. Die Farbe eines 8x8-Blocks resultiert aus den oberen 4 Bits des Video-RAMs.

2.3.3 60x200-Multicolormode (MC):

Allgemeines:

Diese Graphikart ist ähnlich dem HGR-Modus. Hier allerdings bilden je 2 Bits des Bildspeichers einen doppelt so großen Punkt auf dem Schirm, daher die verminderte x-Auflösung. (Bits: 00=Punkt besitzt Hintergrundfarbe; 01=Farbe 1; 10=Farbe 2; 11=Farbe 3). Wie Sie sehen, haben Sie im MC-Modus eine größere Farbauflösung. Hier können in einem 8x8-Feld also 4 Farben vorkommen.

Das zum Setzen eines Punktes notwendige Koordinatensystem sieht dann folgendermaßen aus:


160x200 - MC - Koordinatensystem
P(71;82)
0
50
100
150
x
0
100
y
200

Speichernutzung:

Zum Bildspeicher ist gleiches zu sagen, wie beim HGR-Modus. Lediglich die Farbrealisierung ist verändert:
Die Hintergrundfarbe kommt nun aus dem niederwertigen Halbbyte (4 Bits); Farbe 2 wird aus dem höherwertigen Halbbyte des Videorams gewonnen. Farbe 3 schließlich stammt aus dem Farb-RAM.

Letzteres wirft einige Probleme im Multicolormodus auf:
Da die Farbe des Textes ebenfalls hier abgelegt wird, kommt es zu Unverträglichkeiten beim Gebrauch von Text in der LGR-Seite, während ein MC-Bild angezeigt wird: Wird zu diesem Zeitpunkt normaler Text per PRINT-Statement geschrieben, so ändert sich an der entsprechenden Stelle im Farbram, also in der Farbgebung des MC-Bildes die Farbe. Gleichzeitig fehlt diese nachher bei der Wiederherstellung des Textbildes. Vermeiden Sie also neuen Text im LGR-Bild, während das MC-Bild angezeigt wird (Die LOW-Graphik bleibt davon jedoch unberührt).

2.4 Bemerkung vor dem Start

Bevor wir richtig in den Befehlsschatz der Supergraphik eintauchen, soll hier noch etwas über die Arbeitsweise mit der Supergraphik gesagt werden. Es handelt sich ja - wie gesagt - um eine BASIC-Erweiterung. D.h. Sie können mit diesem Programm nach dem Start genauso arbeiten, wie unter ganz normalem BASIC. Es werden Ihnen nur noch eine ganze Menge zusätzliche Befehle zur Verfügung gestellt. Um Ihnen nun auch die Arbeit mit ihren Programmen zu vereinfachen, existieren einige Programmierutilities, die jedoch erst am Ende dieses Kapitels erläutert werden, um Sie nicht vorweg schon mit zuviel Theorie zu langweilen. Dort finden Sie Befehle zur Anzeige des Disketteninhaltsverzeichnisses, zur automatischen Umnumerierung der BASIC-Zeilen usw. Eine Möglichkeit der Supergraphik soll hier aber schon einmal kurz angesprochen werden: Sie können die 8 Funktionstasten mit einer beliebigen (max. 16 Zeichen langen) Zeichensequenz belegen. Dort können Sie z.B. einige Befehle oder Befehlskombinationen ablegen, die Sie ständig benötigen. Wie das gemacht wird, erfahren Sie beim Befehl KEY am Ende des Kapitels.

Die Funktionstasten sind aber bereits vorbelegt. Bei dem Betätigen einer dieser Tasten unmittelbar nach einem RETURN werden direkt schon einige Befehle ausgeführt. Hier eine Aufstellung der Vorbelegungen:

```
f1: LIST
f2: Bildschirm löschen + LIST
f3: Disketteninhalt zeigen (ohne Programmverlust)
f4: Graphik-Text-Mischanzeige, LGR befehligt
f5: Graphik ausschalten
f6: Graphik-Text-Mischanzeige, HGR befehligt
f7: Graphik ausschalten
f8: Zeilen umnumerieren (RETURN erforderlich)
```

Probieren Sie es doch einfach einmal aus: Betätigen Sie eine der Funktionstasten und dann RETURN.

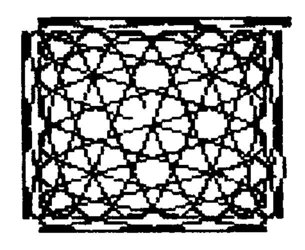
2.5 Alles begann beim Punkt

Als der große griechische Philosoph Euklid vor mehr als 2000 Jahren einmal die Eindrücke einer wunderschönen griechischen Landschaft in den Sand zeichnen wollte und er mit seinem Holzstab zum malen ansetzte, hielt er einen Moment inne, setzte den Stab wieder ab und betrachtete sich den Eindruck, den dieser im Sand hinterlassen hatte. Von diesem Moment an wußte er, daß alles mit dem Punkt begann. Obwohl diese Entdeckung lange Zeit in Vergessenheit geraten war, bekommt sie im Zeitalter der Computer wieder zentrale Bedeutung. Hier sind nämlich alle Graphiken, Bilder und Buchstaben im Endeffekt aus Punkten zusammengesetzt. So, wie also die Materie aus kleinsten Teilchen besteht, so verhält es sich mit der Graphik ebenso. Die glatteste Fläche, die ebenste Linie ist nur eine Annäherung, die unser Auge täuscht. Damit ist ein Befehl, der einen einzelnen Punkt auf dem Bildschirm setzt die Grundlage aller Graphik. Alle anderen Figuren beruhen auf dieser Rechenvorschrift und stellen lediglich eine Erleichterung für uns dar. Wollen wir sehen, wie es sich damit verhält!

2.5.1 PLOT und was dazu gehört

Mit dem Befehl PLOT, dem Befehl, der einen einzigen Punkt auf den Bildschirm zaubert, wollen wir unsere Einführung in die Graphik auf dem Commodore 64 beginnen. Mit diesem Befehl werden Sie eine Menge über die wesentliche Struktur der Graphik und der Supergraphik lernen. Dieses Kapitel sollten Sie also besonders aufmerksam lesen. Zunächst aber probieren Sie doch einmal das folgende Programmbeispiel, das Ihnen das Weiterlesen schmackhaft machen soll. Laden Sie also die Supergraphik wie in Abschnitt 2.2. beschrieben ein und tippen folgendes:

```
100 REM *****************
110 REM **             **
120 REM **  PLOT-DEMO  **
130 REM **             **
140 REM *****************
150 REM
160 GMODE 0,1 : GCLEAR
170 SCOL= 0,0
180 FOR X=0 TO 940 STEP 0.4
190 PLOT ,35*SIN(X/15)+160,40*SIN(X/50)+100
200 NEXT X
210 REM
220 FOR X=0 TO 940 STEP 0.4
230 PLOT ,40*SIN(X/50)+160,35*SIN(X/15)+100
240 NEXT X
```



Sie müssen nicht unbedingt verstehen, was es mit den merkwürdig erscheinenden Formeln in diesem Programm auf sich hat. Es handelt sich hier um eine Form der sogenannten Lissajous-Kurven. Diese kleine Demonstration sollte Sie nur auf das Thema einstimmen. Sprechen wir das Programm doch einmal kurz durch:

In Zeile 160 wird die hochaufgelöste Graphik durch den Befehl GMODE eingeschaltet (wir kommen später noch einmal darauf zurück) und der Graphikbildschirm gelöscht (GCLEAR). Anschließend legen wir in Zeile 170 die Hintergrundfarbe durch SCOL= auf schwarz fest. Nun folgt das eigentliche Zeichenprogramm. Durch die FOR...NEXT-Schleife wird die Variable X ständig um 0,4 erhöht, die in die Formeln zur Koordinatenberechnung des PLOT-Befehls einfließt. Um den PLOT-Befehl, den Befehl also, der einen einzigen Punkt auf den Bildschirm zeichnet, näher zu verstehen, schauen wir ihn uns doch einmal etwas genauer an:

PLOT

Befehl:	PLOT zm,x,y
Beispiel:	PLOT 1,100,199
Parameter:	zm: Zeichenmodus
	x: x-Koordinate
	y: y-Koordinate
Funktion:	Setzen (Löschen etc.) eines Punktes

Dieser Befehl ermöglicht Ihnen also, einen Punkt mit den Koordinaten x,y auf dem Bildschirm in der entsprechenden Farbe (s.u.) zu zeichnen. Um das mit den Koordinaten zu verstehen, rufen wir uns noch einmal das Supergraphik-interne Koordinatensystem in Erinnerung, das bereits in Kapitel 2.3. vorgestellt wurde. In diesem Koordinatensystem lag der Nullpunkt ja links oben in der Ecke des Bildschirmfensters. Die Koordinaten der y-Achse wurden nach unten(!) hin, die der x-Achse nach rechts immer größer. Dabei kamen folgende Werte für x und y in Frage, wenn ein Punkt innerhalb des Bildschirms liegen sollte:

LGR:	x: 0 bis 79
	y: 0 bis49
HGR:	x: 0 bis 319
	y: 0 bis 199
MC:	x: 0 bis 159
	y: 0 bis 199

Dabei bedeuten:

LGR:	Low-graphic	(niedrig aufgelöste Graphik)
HGR:	High-graphic	(hoch aufgelöste Graphik)
MC:	Multicolor	(Multicolor-Graphik)

In unserem Beispiel haben wir es mit HGR zu tun. Damit befindet sich der Punkt mit den Koordinaten x=0 und y=199 in der unteren linken und der Punkt mit den Koordinaten x=319 und y=199 in der unteren rechten Ecke des Bildschirms. In dem Befehl PLOT zm,x,y geben Sie also für x die x- und für y die y-Koordinate des Punktes an, der gezeichnet werden soll. Statt diese Werte direkt mit Zahlen anzugeben, können Sie natürlich dafür auch jede Variable oder Formel einsetzen, die Ihnen in den Sinn kommt, wie dies auch in dem obigen Beispiel geschehen ist: z.B.

```
PLOT 0,(x+1)*5,b^2
```

```
10 GMODE 0,1 : GCLEAR
20 PLOT 0,159,199
30 WAIT 198,255 : REM AUF TASTE WARTEN
```

Dieses Programm zeichnet einen einzelnen Punkt in die Mitte des Bildschirms. Bei einem Tastendruck kehrt das Programm automatisch in den Textmodus zurück. Es zeigt Ihnen die Grundkonfiguration eines Graphikprogrammes auf: Graphik ein, löschen und zeichnen (Zeile 30 ist nur dazu da, damit die Graphik nach dem Setzen des Punktes eingeschaltet bleibt).

Noch eine Wichtigkeit am Rande: Bei allen Befehlen, auch allen späteren Kommandos der Supergraphik ist ein allgemeiner Punkt zu beachten: Folgt einer IF...THEN-Konstruktion ein Befehl des Supergraphik-Befehlsvorrates, so ist nach dem "THEN" unbedingt ein ":" (Doppelpunkt) zu setzen, also zum Beispiel:

```
10 IF A=3 THEN : PLOT 0,100,50
```

Einen Parameter des PLOT-Befehls haben wir in unseren Betrachtungen zunächst noch gar nicht untersucht, den Zeichenmodus zm. Mit diesem Zeichenmodus hat es seine besondere Bewandtnis. Er ist mit dafür verantwortlich, daß jeder einzelne einer ganzen Reihe von Befehlen eine Vielzahl von Ausführungsmöglichkeiten besitzt. Der Zeichenmodus wird uns also auf Schritt und Tritt begleiten. Schauen wir uns doch erst einmal im Überblick an, was unter "Zeichenmodus" zu verstehen ist:

Der Zeichenmodus gibt an, wie die jeweiligen Figuren (in diesem Fall ein Punkt) angesprochen werden sollen. Die folgende Tabelle gibt Ihnen die Werte an, die Sie für zm einsetzen können. Für größere Werte oder beim Weglassen von zm (s.u.) wird automatisch der Wert 0 eingesetzt:

zm	Wirkung	Beispiel
0	Figur zeichnen	PLOT 0,160,100
1	Figur löschen	PLOT 1,160,100
2	Figur invertieren	PLOT 2,160,100
3	Figur punktieren	PLOT 3,160,100
4	Graphikcursor bewegen	PLOT 4,160,100

Wichtig ist, daß unbedingt nach der Angabe des Zeichenmodus ein Komma gesetzt wird (auch z.B. bei PLOT 0,TO 20,50 - auf diesen Befehl werden wir weiter unten zu sprechen kommen). Der Befehl PLOT ,160,100 ersetzt den Befehl PLOT 0,160,100 (schnellere Bearbeitung / speicherplatzsparend).

zu zm=0:

Bisher haben wir nur Fälle kennengelernt, in denen ein Punkt tatsächlich auf den Bildschirm gezeichnet werden sollte. In diesem Fall setzten wir für den Zeichenmodus den Wert 0 ein. In unserem ersten Beispiel haben wir ihn sogar weggelassen. Dies ist tatsächlich erlaubt. Schreiben wir statt PLOT 0,160,100 nur einfach PLOT ,160,100, so wird exakt das gleiche ausgeführt, vorausgesetzt das Komma direkt hinter PLOT wird gesetzt (ansonsten erscheint ein SYNTAX ERROR). Dieser Befehl ist kürzer und auch schneller.

zu zm=1:

Hier haben wir unser erstes Beispiel, bei dem wir einen anderen Zeichenmodus gewählt haben. Schauen wir es uns doch einmal an:

```
100 REM ***************
110 REM **           **
120 REM **  BLINKER  **
130 REM **           **
140 REM ***************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR Z=1 TO 20
180   PLOT 0,159,99 : PLOT 0,160,99
190   FOR X=1 TO 200 : NEXT X
200   PLOT 1,159,99 : PLOT 1,160,99
210   FOR X=1 TO 200 : NEXT X
220 NEXT Z
```

Nach den üblichen Formalitäten in Zeile 10 (wir kommen - wie gesagt - später darauf zurück), werden in der äußeren FOR...NEXT-Schleife zwei Punkte nebeneinander mitten auf dem Bildschirm gezeichnet (Zeile 30) und nach einer kurzen Warteschleife (Zeile 40) wieder in

Zeile 50 gesetzt, indem jetzt die beiden Punkte mit denselben Koordinaten nur mit einem anderen Zeichenmodus angesprochen werden. Auf diese Weise blinkt der Punkt zwanzigmal, um dann wieder hinter dem erscheinenden Textfenster ganz zu verschwinden.

Das folgende Programm demonstriert Ihnen eine weitere kleine Anwendung:

```
100 REM ************************************
110 REM **                                **
120 REM **  DER RASENDE STECKNADELKOPF    **
130 REM **                                **
140 REM ************************************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 Z=0
180 FOR Y=0 TO 199
190   PLOT 0,50+Y,Y : PLOT 0,51+Y,Y
200   PLOT 1,50+Z,Z : PLOT 1,51+Z,Z
210   Z=Y
220 NEXT Y
```

In diesem Programm werden zwei nebeneinander liegende (der besseren Sichtbarkeit wegen) Punkte auf einer Diagonalen bewegt. Dabei werden sie bei jedem Schleifendurchlauf der FOR...NEXT-Schleife in den Zeilen 180-220 einen Punkt weiter gezeichnet (Zeile 190). Dann werden die in dem vorhergehenden Schleifendurchlauf gezeichneten Punkte wieder gelöscht. Dadurch entsteht der Eindruck der Bewegung auf dem Bildschirm. Sie können die beiden Punkte jetzt natürlich auf alle möglichen anderen Bahnen schicken oder selbst per Tastendruck steuern.

Zu zm=2:

Dieser Modus erweitert Ihre Möglichkeiten enorm. Die Figur wird normal gezeichnet. An den Stellen jedoch, an denen vorher ein Punkt gesetzt war, wird dieser Punkt jetzt gelöscht und umgekehrt (Invertierung). Anmerkung: Im Multicolormodus findet ein Farbwechsel statt:

```
Farbe 0 wechselt nach Farbe 3
Farbe 1 wechselt nach Farbe 2
Farbe 2 wechselt nach Farbe 1
Farbe 3 wechselt nach Farbe 0
```

wobei unter Farbe 0 jeweils die Hintergrundfarbe gemeint ist.)

Auf den ersten Blick mag diese Funktion als reine Spielerei erscheinen. Der Vorteil des invertierten Zeichnen liegt darin, daß nach jeweils zweimaligem Zeichnens der vorherige Zustand des Bildes wiederhergestellt ist. Angenommen, an der Stelle, an der gezeichnet werden soll, steht bereits ein Punkt. In diesem Falle wird ja nach der Invertierungsvorschrift dieser Punkt gelöscht. Beim nächsten Invertieren dieses Punktes wird er aber wieder gesetzt und es ist damit alles wieder beim alten. Entsprechendes gilt, wenn an der Stelle vorher kein Punkt gesetzt war. Dieser Effekt wird ein paar Zeilen weiter unten in dem ersten Teilstück unseres großen CAD-Zeichenprogrammes genutzt, um einen blinkenden Cursor auf dem Bildschirm darzustellen, der jedoch das bereits gezeichnete Bild nicht zerstört. Aber lesen Sie ruhig weiter.

Zunächst wollen wir nämlich diesen Befehl an einem kleinen Beispiel kennenlernen:

```
100 REM ********************
110 REM **                **
120 REM **  INVERTIERUNG  **
130 REM **                **
140 REM ********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR ZM=0 TO 2 STEP 2     :REM NORMAL/INVERS-SCHLEIFE
180   FOR X=20 TO 100 STEP 3
190     FOR Y=20 TO 100      :REM SENKRECHTE LINIEN
200       PLOT ZM,X+V,Y      :REM PLOT NORMAL ODER INVERS
210     NEXT Y
220   NEXT X
230   REM
240   FOR Y=20 TO 100 STEP 3
250     FOR X=20 TO 100      :REM WAAGERECHTE LINIEN
260       PLOT ZM,X+V,Y      :REM PLOT NORMAL ODER INVERS
270     NEXT X
280   NEXT Y
290   V=100                  :REM INVERS 100 PUNKTE VERSCHOBEN
300 NEXT ZM                  :REM JETZT INVERS ZEICHNEN
```

In diesem Programm wird zunächst ein Feld sich schneidender senkrechter und waagerechter Linien durch normales Zeichnen auf den Bildschirm gebracht. Im Anschluß daran (gesteuert durch die äußerste FOR...NEXT...-Schleife) wird das gleiche Objekt noch einmal, diesmal aber im inversen Modus gezeichnet. Wenn Sie das Programm testen, wird Ihnen der Unterschied sofort ins Auge fallen: Die Schnittpunkte der Linien des zweiten Feldes wurden durch die Invertierung wieder gelöscht.

zu zm=3

Nun eine kleine Spielerei: Sie besitzen die Möglichkeit, jede Figur (außer Punkten natürlich) punktiert zu zeichnen, d.h. es wird nur jeder zweite eigentlich zu zeichnende Punkt gesetzt (Hierdurch entstehen schöne Effekte: in Feld wird beispielsweise längsgestreift gezeichnet, was seltsame Farbereignisse bewirkt. Ein Rahmen dagegen erhält ein besonderes Aussehen: das eines echten Bilderrahmens!). Da dieser Zeichenmodus beim Setzen von Punkten keinen Effekt bewirkt, werden wir ihn später noch einmal erwähnen. Hier sei er nur der Vollständigkeit halber genannt.

zu zm=4:

Ihre Supergraphik 64 merkt sich jeweils den zuletzt gezeichneten Punkt und behält ihn als imaginären Graphikcursor, um ihn bei einigen Zeichen-Befehlen statt Ihrer Koordinatenangabe einzusetzen, so daß Sie direkt an andere Figuren (z.B. Kreisbögen oder Linien) anknüpfen können. Setzen Sie zm nun gleich 4, so wird lediglich dieser unsichtbare Graphikcursor bewegt, ohne daß sich sonst irgendwelche Veränderungen auf dem Bildschirm ergeben. Auch auf diesen Befehl werden wir später noch (beim Zeichnen von Linien und beim T-Zusatz der erweiterten Syntax) zurückkommen.

2.5.2 CAD-Zeichner, das Zeichenpaket

Soweit zu der einfachen Anwendung. Jetzt aber wollen wir uns endlich dem ersten großen Teil unseres CAD-Zeichenprogrammes widmen. Wir stellen hier das Grundgerüst mit einigen bereits realisierten Funktionen vor, die wir mit unseren bisherigen bescheidenen Mitteln jetzt schon realisieren können.

```
100 REM ********************
110 REM **                **
120 REM **  CAD-ZEICHNER   **
130 REM **                **
140 REM ********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 X=160 : Y=100           :REM KOORDINATEN INIT
175 XA=320 : YA=200         :REM X- UND Y-AUFLOESUNG
180 POKE 650,255            :REM AUF ALLE TASTEN REPEAT
190 REM
200 REM EINGABESCHLEIFE:
210 REM
220 FL=0                    :REM FLAG ZURUECKSETZEN
230 PLOT 2,X,Y : PLOT 2,X+1,Y:REM PUNKT INVERTIEREN
240 FL=ABS(FL-1)            :REM FLAG 0<->1 WECHSELN
250 FOR Z=1 TO 30
260 GET A$ : IF A$="" THEN NEXT Z : GOTO 230
270 IF FL=1 THEN : PLOT 2,X,Y : PLOT 2,X+1,Y :REM PUNKT WIEDER ZURUECKSE
TZEN
280 REM
290 REM
300 REM VERZWEIGUNG:
310 REM
320 A=ASC(A$)
330 IF A=17  THEN GOSUB 1300 :REM RUNTER
340 IF A=145 THEN GOSUB 1240 :REM HOCH
350 IF A=29  THEN GOSUB 1340 :REM RECHTS
360 IF A=157 THEN GOSUB 1390 :REM LINKS
370 IF A=133 THEN GOSUB 1010 :REM F1
380 IF A=137 THEN GOSUB 1040 :REM F2
390 IF A=134 THEN GOSUB 1070 :REM F3
400 IF A=138 THEN GOSUB 1100 :REM F4
410 IF A=135 THEN GOSUB 1130 :REM F5
420 IF A=139 THEN GOSUB 1160 :REM F6
430 IF A=136 THEN GOSUB 1190 :REM F7
440 IF A=140 THEN GOSUB 1220 :REM F8
450 IF A= 19 THEN CF=ABS(CF-1) :REM SCHNELL/LANGSAM
460 IF A$="Q" THEN:GMODE 0,0 : END
900 GOTO 220
960 REM
970 REM
980 REM AUSFUEHRUNG:
990 REM
1000 REM F1: PUNKT SETZEN UND RECHTS
1010 PLOT ,X,Y : GOTO 1340
1020 REM
1030 REM F2: PUNKT LOESCHEN UND RECHTS
1040 PLOT 1,X,Y : GOTO 1340
1050 REM
```

```
1060 REM F3: PUNKT SETZEN UND LINKS
1070 PLOT ,X,Y : GOTO 1390
1080 REM
1090 REM F4: PUNKT LOESCHEN UND LINKS
1100 PLOT 1,X,Y : GOTO 1390
1110 REM
1120 REM F5: PUNKT SETZEN UND HOCH
1130 PLOT ,X,Y : GOTO 1240
1140 REM
1150 REM F6: PUNKT LOESCHEN UND HOCH
1160 PLOT 1,X,Y : GOTO 1240
1170 REM
1180 REM F7: PUNKT SETZEN UND RUNTER
1190 PLOT ,X,Y : GOTO 1300
1200 REM
1210 REM F8: PUNKT LOESCHEN UND RUNTER
1220 PLOT 1,X,Y : GOTO 1300
1230 REM
1240 REM CURSOR HOCH:
1250 IF CF=1 THEN Y=Y-9
1260 Y=Y-1 : IF Y<0 THEN Y=YA+Y
1270 RETURN
1280 REM
1290 REM CURSOR RUNTER:
1300 IF CF=1 THEN Y=Y+9
1310 Y=Y+1 : IF Y>=YA THEN Y=Y-YA
1320 RETURN
1330 REM
1340 REM CURSOR RECHTS:
1350 IF CF=1 THEN X=X+9
1360 X=X+1 : IF X>=XA THEN X=X-XA
1370 RETURN
1380 REM
1390 REM CURSOR LINKS:
1400 IF CF=1 THEN X=X-9
1410 X=X-1 : IF X<0 THEN X=XA+X
1420 RETURN
```

Gehen wir das Programm doch einmal durch: Zunächst einmal können Sie sich auf dem Graphikbildschirm mit einem kleinen, blinkenden Zwei-Punkte-Cursor mithilfe der Cursortasten frei bewegen. Dabei stellt die Eingabeschleife (Zeilen 220-260) das blanke Grundgerüst dar. In dieser Schleife wird - neben dem Blinken des Cursors (Zeile 230) - ständig auf einen Tastendruck getestet (Zeile 260). Ist dieser Tastendruck erfolgt, so wird die Zeile 270 abgearbeitet, die den ursprünglichen Zustand des Punktes unter dem Cursor wieder herstellen soll. Hierzu wird das Flag FL verwendet, das synchron mit dem Cursorblinken ständig zwischen 0 und 1 wechselt. Dieses Wechseln wird

durch die Formel in Zeile 240 bewirkt: Ist FL=0, so wird ABS(FL-1) eins, ist FL=1, so nimmt die Formel den Wert 0 an. In Zeile 270 wird nun getestet, ob der Cursor zur Zeit ein- oder ausgeschaltet ist und gegebenenfalls ausgeschaltet.

Was nun folgt, ist die Auswertung des erfolgten Tastendrucks. Nacheinander wird auf die einzelnen erlaubten Tasten getestet und zu der entsprechenden Routine gesprungen (Zeilen 330-460). Die Ausführung dieser Routinen beginnt ab Zeile 1000. Im einzelnen haben wir folgenden Tasten folgende Funktionen zugeordnet. Diese Zuordnung kann selbstverständlich jederzeit von Ihnen geändert werden:

CRSR hoch	- bewegt blinkenden Punkt hoch
CRSR runter	- bewegt blinkenden Punkt hinunter
CRSR rechts	- bewegt blinkenden Punkt nach rechts
CRSR links	- bewegt blinkenden Punkt nach links
F1	- Punkt setzen und CRSR rechts
F2	- Punkt löschen und CRSR rechts
F3	- Punkt setzen und CRSR links
F4	- Punkt löschen und CRSR links
F5	- Punkt setzen und CRSR hoch
F6	- Punkt löschen und CRSR hoch
F7	- Punkt setzen und CRSR hinunter
F8	- Punkt löschen und CRSR hinunter
clr/home	- Wechsel zwischen schnell und langsam
Q	- Beenden des Programms

Unter "Wechsel zwischen schnell und langsam" ist dabei die Anzahl der Punkte zu verstehen, mit denen sich der Cursor bei einem Tastendruck fortbewegt. Dies können entweder 1 oder 10 Punkte sein.

Beachten Sie hier auch die Anwendung der Regel, die einen Doppelpunkt zwischen THEN einer IF...THEN-Anweisung und einem Befehl der Supergraphik vorschreibt: Zeilen 270 und 460!

Mit diesem Programm können Sie bereits aus der Hand - wenn auch noch etwas umständlich - eigene Bilder zeichnen. Schauen Sie sich das Programm ruhig einmal etwas genauer an und nehmen, wenn Ihnen etwas nicht gefällt, Änderungen vor. In den nächsten Abschnitten werden wir - wie gesagt - dieses Grundgerüst noch um einige schöne Funktionen erweitern.

2.5.3 PRINT ist out

Sie kennen sicher die Möglichkeit Ihres Commodore 64, einfache Graphiken im normalen Text-Modus mithilfe der Graphiksymbole des Commodore-Zeichensatzes zu erzeugen. Wie Sie später noch erfahren werden (Kapitel 5.1.), gibt es dafür im wesentlichen drei verschiedene Möglicheiten, die sich der beiden Befehle PRINT und POKE bedienen. Der Vorteil von PRINT ist dabei die Möglichkeit der direkten Eingabe der Graphikzeichen ins Programm und der resultierenden Programmkürze. Großer Nachteil aber: Die jeweilige Bildschirmposition kann nur sehr umständlich mit vielen PRINTs und TABs erreicht werden. Der Nachteil der POKE-Methode: Sehr umständliche Zeicheneingabe und Adressenberechnung jeder Bildschirmposition.

Mit der Supergraphik werden Ihnen nun zwei wesentlich verbesserte Möglichkeiten in die Hand gegeben, die die Erstellung solcher Graphiken in hohem Maße vereinfachen und rationalisieren. Zunächst zur ersten Methode:

2.5.3.1 Positionslichter

Supergraphik bietet Ihnen einen sehr interessanten Befehl namens POS= (nicht zu verwechseln mit dem Befehl "POS(", der innerhalb eines PRINT-Statements verwendet wird). Mithilfe dieses Befehles ist es Ihnen möglich, den normalen Textcursor an einer beliebigen Stelle des Bildschirms zu positionieren. Wenn Sie nun mit PRINT etwas ausgeben, so beginnt die Ausgabe exakt an dieser Stelle:

POS=

Befehl:	POS= s,z	
Beispiel:	POS= 10,15	
Parameter:	s:	Spalte (0-39)
	z:	Zeile (0-24)
Funktion:	Positionieren des Text-Cursors	

Das folgende Programm soll Ihnen einen kleinen Einblick in die mannigfaltigen Anwendungsmöglichkeiten dieses Befehls geben:

```
100 REM ************************
110 REM **                    **
120 REM **  POSITIONSLICHTER  **
130 REM **                    **
140 REM ************************
150 REM
160 PRINT CHR$(147) : REM BILDSCHIRM LÖSCHEN
170 FOR X=1 TO 500
180   POS= RND(1)*40,RND(1)*25 : REM ZUFALLSPOSITION
190   PRINT "HALLO!";
200 NEXT X
```

Hier wird, wie Sie sehen, nicht in Graphik umgeschaltet, sondern im normalen Textmodus operiert. Dazu wird zunächst in Zeile 160 der gesamte Bildschirm gelöscht (shift clr/home). Dann wird 500 mal an zufällige Positionen "HALLO!" geschrieben. Die zufälligen Positionen werden wie folgt berechnet: Die Funktion RND(1) übergibt stets Zufallswerte im Bereich von 0-1. Durch Multiplikation mit 40 (bzw. 25) wird dieser Wertebereich auf 0-40 (bzw. 0-25) erweitert. Damit bekommen wir gültige Werte für Spalte und Zeile zur Cursorpositionierung. Anmerkung: Diese Technik kann natürlich auch zur zufälligen Punktsetzung in der hochauflösenden Graphik verwendet werden.

2.5.3.2 Statische und ...

Kommen wir nun zu der zweiten Möglichkeit der Graphikerzeugung in LGR mit der Supergraphik:
Und da sind wir auch schon wieder bei unserem PLOT-Befehl. Denn PLOT kann nicht nur in der hochauflösenden Graphik (und der Multicolorgraphik selbstverständlich) verwendet werden. In LGR, also im normalen Textmodus wurde eine weitere Graphikauflösung realisiert, die 50x80-Punktegraphik mit allen Vorteilen des Textmodus.

Unter Paragraph 2.3.1. können Sie in aller Kürze nachlesen, was es mit der niedrig aufgelösten Graphik (LGR) auf sich hat. Grundsätzlich können sie mit ihr fast genauso umgehen (gleiche Zeichenbefehle etc.), wie mit der bereits bekannten Graphik. Sie brauchen nicht einmal einen besonderen Graphikmodus oder ähnliches einzuschalten. Der PLOT-Befehl merkt automatisch, daß er jetzt im Textmodus zeichnen soll (daß es doch noch etwas anders ist, lesen Sie im Abschnitt 2.5.3. nach). Die von ihm auf den Bildschirm gebrachten Punkte werden nur dort gezeichnet, wo sich kein Text befindet, um nichts unnötig zu zerstören. Doch sehen Sie selbst:

```
100 REM ******************
110 REM **              **
120 REM **  SINUSKURVE  **
130 REM **              **
140 REM ******************
150 REM
160 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
170 POS= 12,10 : PRINT "SINUSKURVE"
180 REM
190 REM
200 REM Y-ACHSE ZEICHNEN:
210 REM
220 Z=3                :REM OBERER STARTWERT
230 FOR Y=0 TO 49 STEP 6
240   FOR T=1 TO 6     :REM ALLE 6 PUNKTE EIN TEILSTRICH
250     PLOT ,8,Y+T    :REM Y-ACHSE
260   NEXT T
270   PLOT ,7,Y+T      :REM TEILSTRICH
280   IF Z<0 THEN:PLOT B 45,,3,Y+T :REM MINUSZEICHEN
290   PLOT B 48+ABS(Z),,5,Y+T :REM ZIFFER
300   Z=Z-1            :REM NAECHSTE ZIFFER
310 NEXT Y
320 REM
330 REM
340 REM X-ACHSE ZEICHNEN:
350 REM
360 Z=0                    :REM LINKER STARTWERT
370 FOR X=0 TO 79 STEP 8 :REM ALLE 8 PUNKTE EIN TEILSTRICH
380   FOR T=1 TO 8
390     PLOT ,X+T,25       :REM X-ACHSE
400   NEXT T
410   PLOT ,X+T,26         :REM TEILSTRICH
420   IF Z<0 THEN:PLOT B 45,,X+T-1,28 :REM MINUS
430   PLOT B 48+ABS(Z),,X+T,28 :REM ZIFFER
440   Z=Z+1                :REM NAECHSTE ZIFFER
450 NEXT X
460 REM
470 REM
480 REM SINUSKURVE ZEICHNEN:
490 REM
500 FOR X=0 TO 79 STEP 0.5
510   PLOT ,X,20*SIN(X/10)+25
520 NEXT X
530 REM
540 WAIT 198,255
```

Wenn Sie dieses Programm laufen lassen, wird Ihnen eine schöne Sinuskurve mit Koordinatenachsen und Beschriftung ausgegeben. Gehen wir das Programm doch einmal durch:

Bevor wir beginnen, im normalen Textmodus zu zeichnen, löschen wir zweckmäßigerweise zunächst einmal den gesamten Bildschirm, wie dies in Zeile 160 geschieht. Um Ihnen nun zu demonstrieren, daß bereits vorhandener Text nicht von der Punktgraphik überschrieben wird, schreiben wir mitten in den Bildschirm (POS=-Befehl, s.o.) den Titel "Sinuskurve".

Dann beginnt die Konstruktion der Koordinatenachsen. (Wenn Ihnen das ganze zu kompliziert ist, was bei diesem Programm durchaus der Fall sein kann, überlesen Sie einfach diesen Absatz). Zu diesem Zwecke konstruieren wir in den Zeilen 230-310 eine FOR...NEXT-Schleife, die von oben nach unten die gesamte y-Achse zeichnet. Innerhalb dieser Schleife befindet sich eine zweite (Zeilen 240-260), die jeweils 6 Punkte der Achse zeichnet. Alle 6 Punkte wird dann in Zeile 270 ein Teilstrich gezeichnet und in den Zeilen 280/290 mit einer Ziffer beschriftet. Hier taucht auch gleich eine Besonderheit auf, die wir erklären müssen:

Alle Befehle nämlich, die die Angabe des Zeichenmodus verlangen (also auch unser PLOT), können zusätzlich durch drei weitere Funktionen (wahlweise einzeln oder kombiniert) erweitert werden. Dies geschieht durch eine Erweiterung der Befehlssyntax. Wir wollen von diesen drei Erweiterungsmöglichkeiten zunächst die eine hier angewandte besprechen. Die beiden anderen folgen in anderen Paragraphen:

B m - Pinselwahl
Beispiel: PLOT B 45,,20,40

An dem obigen Beispiel erkennen Sie die Anwendung dieser Syntaxerweiterung. Der entsprechende Buchstabe (hier "B") wird einfach an das eigentliche Befehlswort (mit oder ohne Leerzeichen) angehängt. In diesem Fall wird durch diese Syntaxerweiterung ein zusätzlicher Parameter m erforderlich. Es wird die Angabe einer Zahl zwischen 0 und 255 für m gefordert. Diese Zahl hat nun folgende Bedeutung:

In niedrigauflösender Graphik:

In LGR bestimmt diese Zahl den Bildschirmcode eines Zeichens. Dieses Zeichen wird dann statt der sonst verwendeten Punkte in das Textfenster geschrieben. Da das Zeichen natürlich eine Größe von 2x2 LGR-Punkten besitzt, muß man im Programm entsprechende Vorkehrungen treffen. Auf diese Weise ist es also auch möglich, statt mit Punkten, mit einzelnen Zeichen zu malen, ohne vom PRINT-Befehl Gebrauch zu machen. Ein schönes Beispiel für diese Technik ist Ihnen bereits aus dem obigen Programm bekannt. Die Bildschirmcodes (nicht zu verwechseln mit den ASCII-Codes) aller Commodore-Zeichen erfahren Sie aus dem Anhang des normalen C64-Handbuchs.

In hochauflösender (Multicolor-) Graphik:

In HGR bzw. MC bestimmt die Zahl m die Maske für jeden Punkt. Wird also ein Punkt gesetzt, so wird nicht der einzelne Punkt angesprochen, sondern in das betreffende Byte des Graphikspeichers dieser eine Wert eingegeben. Betrachtet man diese Zahl dann als Binärzahl (s. Anhang), also als eine Folge von Nullen und Einsen, geben die 8 Bits dieser Zahl dann also an, welcher Punkt gesetzt, welcher nicht gesetzt werden soll. Am besten sieht man den Effekt, wenn man etwas herumprobiert:

```
10 GMODE 0,1 : GCLEAR : SCOL= 0,0
20 FOR M=1 TO 255
30   FOR X=1 TO 319
40     PLOT B M,,X,X/2
50   NEXT X
60   GCLEAR
70 NEXT M
```

Sehr schöne Effekte bringt diese Befehlserweiterung auch bei den anderen Zeichenbefehlen (CIRCLE, FRAME, FILL, ...), die wir später noch durchsprechen werden.

Doch sollten wir uns nach diesem kurzen Intermezzo wieder dem Sinusprogramm in LGR zuwenden, das wir noch nicht ganz zuende besprochen haben. Die Funktion des B-Zusatzes ist nun geklärt. Damit sollten Sie die Möglichkeit haben, zu verstehen, wie die Beschriftung der Achsen realisiert wurde. Da natürlich stets nur ein Zeichen gleichzeitig mit dem PLOT B-Befehl auf den Bildschirm gebracht

werden kann, werden negative Werte von Z (der Variablen, die die aktuelle Beschriftungszahl enthält) in Zeile 280 abgefangen. Dort wird dann das Minus-Zeichen (Bildschirmcode: 45) an die entsprechende Stelle geschrieben.

Die Zeilen 360-450 bewirken exakt das gleiche wie gerade beschrieben, nur für die x-Achse. Das eben Gesagte kann also hierauf übertragen werden.

Die eigentliche Sinuskurve wird lediglich in den 3 Zeilen 500-520 gezeichnet. Etwas Allgemeines zur Zeichnung von Sinuskurven, also zu der Formel, die in Zeile 510 Verwendung findet: Da eine Sinusfunktion f=sin(x) nur Werte von -1 bis 1 liefert, müssen diese Werte für die Graphik vergrößert werden. Dies geschieht in unserem Falle durch den Streckungsfaktor 20. Dadurch nimmt die Funktion nun Werte zwischen -20 und 20 an. Negative Werte sind in der Graphik aber nicht zulässig, also verschieben wir die ganze Kurve durch ein "+25" in den positiven Bereich. Die Funktion nimmt nun also Werte zwischen 5 und 45 an. Eine Sinuskurve hat einen periodischen Verlauf. Alle 2*pi Punkte wiederholt sich also der Kurvenverlauf. 2*pi sind aber nur ca. 6. Um diese Periode nun zu verlängern, teilen wir alle x-Werte noch einmal durch 10. Hierdurch hat unsere Sinusfunktion nun eine Periode von ca. 60, was sich gut auf dem Graphikbildschirm darstellen läßt. Verändern Sie ruhig einmal diese Werte und Sie werden feststellen, daß sich z.B. die Periode verändert, die Amplitude (also der Ausschlag der Sinuswellen nach oben und unten) oder die ganze Kurve verschiebt. Die hier erläuterten Grundsätze gelten natürlich auch für die hochauflösende Graphik. Hier bieten sich aber dann natürlich insgesamt größere Werte an.

2.5.3.3 ... bewegte Bilder

Im letzen Abschnitt haben wir die verschiedenen Techniken kennengelernt, in der niedrig aufgelösten Graphik mit der Supergraphik schöne Bilder herzustellen. Hier wird Ihnen nun anhand eines kleinen Beispielprogrammes gezeigt, wie Sie bewegte Graphiken leicht realisieren können. In unserem Fall wollen wir demonstrieren, wie in BASIC sogenannte Shapes, also softwaremäßig dargestellte kleinere Graphiken in der Graphik realisiert werden können. In Assembler geht das natürlich alles viel schneller, weshalb die Supergraphik auch dafür (etwas weiter unten) Befehle zur Verfügung stellt. Hier soll aber der Spezialfall einer bewegten Graphik mit allen Commodore-Graphikzeichen demonstriert werden. Schauen Sie sich doch einmal das folgende Programm an:

```
100 REM ********************
110 REM **                **
120 REM **  SPRINGTEUFEL  **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
170 REM
180 REM
190 REM DEFINITIONEN AUS DATAS UEBERNEHMEN:
200 REM
210 DIM A(2,11)       :REM DEFINIERE SPEICHERPLATZ
220 FOR MN=0 TO 2
230 FOR MD=0 TO 11
240 READ A(MN,MD)     :REM DEFINITIONEN EINLESEN
250 NEXT MD
260 NEXT MN
270 REM
280 REM
290 REM MAENNCHEN ZEICHNEN+BEWEGEN:
300 REM
310 XN=10 : YN=10     :REM KOORDINATEN INIT
320 REM
330 MN=2 : GOSUB 550  :REM LOESCHEN
340 X=XN : Y=YN       :REM NEUE KOORDINATEN UEBERNEHMEN
350 MA=ABS(MA-1)      :REM NUMMER WECHSELN
360 MN=MA : GOSUB 550 :REM MAENNCHEN ZEICHNEN
370 REM
380 REM
390 REM AUF EINGABE WARTEN:
400 REM
410 GET A$ : IF A$="" THEN 410
420 XN=X : YN=Y :REM NEUE KOORDINATEN INIT
430 IF ASC(A$)=29  THEN XN=XN+1 : IF XN>35 THEN XN=0
440 IF ASC(A$)=157 THEN XN=XN-1 : IF XN<0  THEN XN=35
450 IF ASC(A$)=17  THEN YN=YN+1 : IF YN>20 THEN YN=0
460 IF ASC(A$)=145 THEN YN=YN-1 : IF YN<0  THEN YN=20
470 REM
480 GOTO 330
490 REM
500 REM
510 REM FIGUR ZEICHNEN:
520 REM UEBERGABE: MN: MAENNCHEN-NR
530 REM           X,Y: MAENNCHEN-KOORD.
540 REM
550 YK=Y
560 FOR MD=0 TO 11 STEP 3
570 POS= X,YK
580 PRINT CHR$(A(MN,MD));CHR$(A(MN,MD+1));CHR$(A(MN,MD+2));
```

```
590 YK=YK+1
600 NEXT MD
610 RETURN
620 REM
630 REM
640 REM MAENNCHEN 1:
650 REM
660 DATA  32,215, 32
670 DATA 205,221,206
680 DATA  32, 84, 32
690 DATA 206,183,205
700 REM
710 REM
720 REM MAENNCHEN 2:
730 REM
740 DATA  32, 32, 32
750 DATA  32,215, 32
760 DATA 206, 84,205
770 DATA 170,183,180
780 REM
790 REM
800 REM LOESCHEN:
810 REM
820 DATA  32, 32, 32
830 DATA  32, 32, 32
840 DATA  32, 32, 32
850 DATA  32, 32, 32
```

Das Prinzip ist das folgende: In einem Array (vorher aus DATA-Zeilen gelesen (Zeilen 220-260)) werden die Definitionen für ein 3x4-Zeichenfeld aus normalen Text-Zeichen bereitgehalten. Was in diesen Definitionen nun festgehalten wird (in diesem Fall die Definitionen für zwei Männchen und ein Leerfeld) ist vollkommen egal. D.h. das übrige Programm braucht in keiner Weise geändert werden, wenn Sie hingehen und die in den DATAs definierten Figuren ändern.

Es existiert nun in den Zeilen 550-610 eine Routine, die stets dann vom Hauptprogramm als Unterroutine aufgerufen werden kann, wenn ein solches 3x4-Feld (wir nennen es in Zukunft immer Shape) auf den Bildschirm gezeichnet werden soll. Dieser Routine werden lediglich die Nummer der betreffenden Shape-Definition und die Koordinaten, an denen diese Definition gezeichnet werden soll, angegeben. Die Unterroutine wird dann zeilenweise dieses Shape ausgeben und durch RETURN (Zeile 610) wieder ins Hauptprogramm zurückspringen.

Dieses Hauptprogramm kann nun auf mannigfaltige Art und Weise gestaltet sein. In diesem Beispiel wird ein Männchen durch Betätigung der Cursor-Tasten auf dem Bildschirm hin- und herbewegt. Dabei bewegt es Arme und Beine. Realisiert wurden diese Effekte durch eine in Zeile 410 programmierte Tastaturabfrage. Wurde eine Taste betätigt, so werden entsprechend den Cursortasten die aktuellen Koordinaten des Shapes geändert (rechts, links, hoch und hinunter). Dies geschieht in den Zeilen 430-460.

Wurden die neuen Koordinaten berechnet, springt das Programm wieder zur Bewegungsroutine, die zunächst einmal das Männchen an der alten Position löscht. Dieses Löschen geschieht durch einfaches Überschreiben des Shapes mit dem Leershape (Zeile 330). Alsdann kann das neue Shape an den neuen Koordinaten neu gezeichnet werden. Beachten Sie bitte, daß zum Zeichnen eines Shapes stets die Unterroutine ab Zeile 550 verwendet wird.

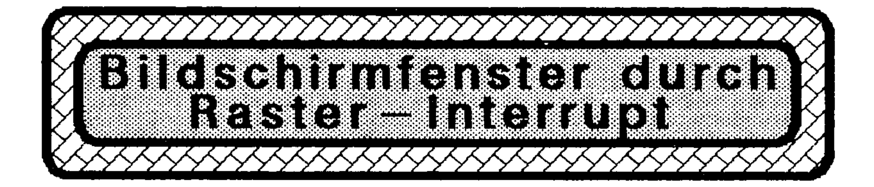
Dabei wird stets abwechselnd Männchen 1 und 2 (bzw. 0 und 1) gezeichnet. Der Nummernwechsel geschieht in Zeile 350. Die Formel in dieser Zeile sollten Sie bereits kennen, wenn Sie das gesamte PLOT-Kapitel bis hierhin gelesen haben.

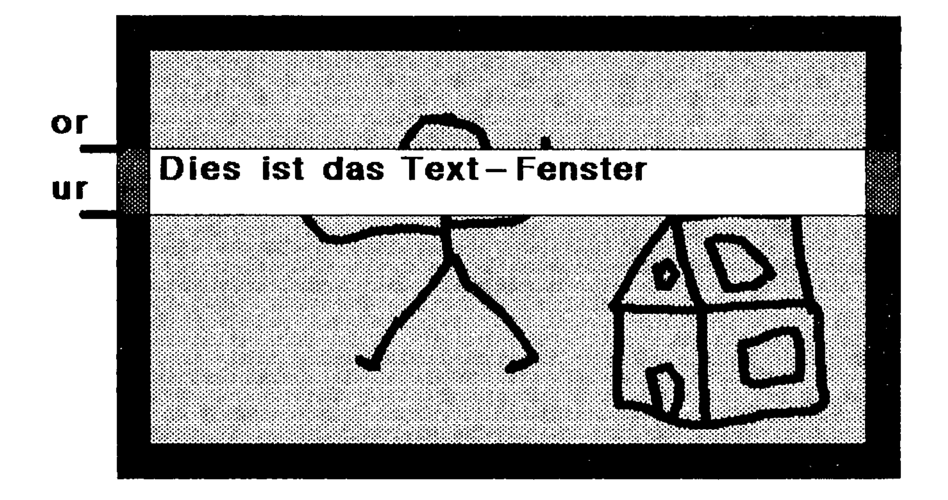
Sie erkennen also das Prinzip der bewegten Graphiken: Die einzelnen Graphiken bewegen sich nicht, es werden vielmehr schnell hintereinander viele unbewegliche, sich geringfügig unterscheidende Bilder gezeigt, die dadurch eine Bewegung simulieren. Wir werden noch des öfteren in diesem Buch auf dieses Prinzip stoßen, das schließlich sozusagen das "Ur"prinzip vieler Spiele darstellt.

2.5.4 Graphikfenster durch Raster-Interrupt

Was Ihren Commodore ebenfalls auszeichnet, ist die geniale IRQ-Steuerung des Betriebssystems. Alle 1/60 Sekunde wird der Prozessor (6510) gestoppt und eine sogenannte IRQ-Routine durchgeführt, um danach wieder an der gleichen Stelle des eigentlichen Programms fortzufahren (dies geschieht natürlich auf der Maschinenspracheebene). Normalerweise dient diese IRQ-Routine z.B. zur Abfrage der Tastatur und Speicherung eventuell gedrückter Tasten (so ist es Ihnen möglich, während Ihres Programmes Engaben bis zu 10 Tastendrücken vorzunehmen, die sich Ihr Rechner merkt und später bearbeitet). Nun ist es möglich, eine eigene IRQ-Routine zu schreiben. In der Supergraphik wurde dies verwirklicht:

So werden die Sprites in der IRQ-Routine bewegt und die Tonsteuerung vorgenommen, während Ihr eigenes Programm weiterläuft. Mithilfe des Raster-IRQs werden gemischte Text- und Graphikanzeige und 16 Sprites auf dem Bildschirm erzeugt.





Was uns in diesem Zusammenhang interessiert, ist die gleichzeitige Darstellung von Text und Graphik auf einem Bildschirm. Wie dies genau in Assembler realisiert wird, soll uns hier nicht interessieren. Falls Sie es doch einmal wissen möchten, so lesen Sie doch einfach in den späteren Kapiteln nach. Dort wird es in aller Ausführlichkeit mit anschaulichen Beispielprogrammen (auch für nicht-Assembler-Programmierer) demonstriert.

Wie können wir nun solche Dinge aber mit unserer Supergraphik realisieren? Nun, dafür ist der bereits altbekannte, aber nie genau erklärte Befehl GMODE zuständig. Dieser Befehl sei nun hier etwas genauer unter die Lupe genommen. Damit Sie sich aber genau vorstellen können, was unter "gleichzeitiger Text- und Graphikanzeige" zu verstehen ist, testen Sie doch erst einmal folgendes Programm (vergessen Sie nicht, vorher die Supergraphik einzuladen):

```
100 REM *********************
110 REM **                 **
120 REM **  RASTER-SPIELE  **
130 REM **                 **
140 REM *********************
150 REM
160 REM LGR-KURVE ZEICHNEN:
170 REM
180 PRINT CHR$(147);
190 FOR X=0 TO 79
200 PLOT ,X,SQR(X*40)
210 NEXT X
220 REM
230 REM
240 REM HGR-KURVE ZEICHNEN:
250 REM
260 GMODE 0,1 : GCLEAR : SCOL= 0,0
270 FOR X=0 TO 319
280 PLOT ,X,90*SIN(X/50)+100
290 NEXT X
300 REM
310 REM
320 REM FENSTER ROLLEN:
330 REM
340 FOR X=1 TO 205 STEP 8
350 GMODE 0,7,X+48,X
360 FOR T=1 TO 10 : NEXT T :REM WARTESCHLEIFE
370 NEXT X
380 GOTO 340
```

Nach dem Zeichnen zweier Kurven - eine in LGR und eine in HGR - läuft ein Graphikfenster über den Bildschirm. Dabei taucht in Zeile 350 eine merkwürdige Form des GMODE-Befehles auf, die wir unter anderem im folgenden erklären wollen. Doch stellen wir GMODE doch erst einmal offiziell vor:

GMODE

	GMODE (s)(,m)(,or,ur)
Beispiel:	GMODE 0,1
oder:	GMODE 2,7,30,70
Parameter:	s: Seitenmodus (s.u.)
	m: Graphikmodus (s.u.)
	or: oberer Textrand bei m=7/8
	ur: unterer Textrand bei m=7/8
Funktion:	bestimmen des Graphikmodus

Schon oft haben Sie sie gebraucht, um von Text auf Graphik umzuschalten und umgekehrt. Nun lernen Sie sie richtig kennen:

Die Graphikmoduswahl

zu s:

Supergraphik 64 besitzt die Möglichkeit, mit 2 unabhängigen Graphikseiten zu arbeiten, d.h. Sie können gleichzeitig neben dem LGR-Bild 2 hochauflösende (bzw. Multicolor) Bilder erstellen und anzeigen (ein Graphikbild nennt man Seite, da man es wie eine Buchseite aufschlagen oder zuklappen kann). Normalerweise bearbeitet man die Seite, die gerade angezeigt wird. Es ist jedoch möglich z.B. die 2. Seite mit denselben Befehlen zu erstellen, während die erste sichtbar ist. Um diese Möglichkeit auszuwählen, geben Sie s (für Seite) an:

```
s=0: Graphikseite 1 wird angezeigt und befehligt
s=1: Graphikseite 1 wird angezeigt, Seite 2 aber befehligt
s=2: Graphikseite 2 wird angezeigt und befehligt
s=3: Graphikseite 2 wird angezeigt, Seite 1 aber befehligt
```

Sie können also Kino spielen:

Während Sie lange Rechnungen und Zeichnungen auf der unsichtbaren Seite ausführen, wird die andere betrachtet, dann schalten Sie um und bearbeiten nun die vorher sichtbare Seite.

```
100 REM *********************
110 REM **                 **
120 REM **  SEITENWECHSEL  **
130 REM **                 **
140 REM *********************
150 REM
160 GMODE 0,1 : GCLEAR :REM SEITE 1 ANZEIGEN + LOESCHEN
170 SCOL= 0,0
180 GMODE 2   : GCLEAR :REM SEITE 2 ANZEIGEN + BEFEHLIGEN
190 SCOL= 0,0
200 REM
210 REM KURVE 2 IN SEITE 2:
220 FOR X=0 TO 319
230   PLOT ,X,(X-160)^2/130
240 NEXT X
250 REM
260 REM KURVE 1 IN SEITE 1:
270 GMOOE 3          :REM SEITE 1 NUR BEFEHLIGEN
280 FOR X=0 TO 319
290   PLOT ,X,EXP(5*SIN(X/40))+50
300 NEXT X
310 REM
320 REM SEITEN WECHSELN:
330 FOR X=1 TO 20
340   GMODE 0          :REM SEITE 1 EIN
350   FOR T=1 TO 200 : NEXT T
360   GMODE 2          :REM SEITE 2 EIN
370   FOR T=1 TO 200 : NEXT T
380 NEXT X
```

Zunächst wird hier Seite 1 eingeschaltet und befehligt (Zeile 160), um gelöscht zu werden. Das gleiche geschieht mit Seite 2 (Zeile 180). In diese zweite Seite wird nun sichtbar eine Kurve eingezeichnet (Zeilen 220-240). Ist dies geschehen, so schaltet Zeile 270 auf die erste Seite um. Dabei bleibt aber die zweite Seite sichtbar, d.h. die Zeichenbefehle gelten von nun ab für die erste Seite. Auf diese Weise wird nun in den Zeilen 280-300 für den Beobachter unsichtbar eine andere Kurve in Seite 1 gezeichnet. Während dieser Zeit scheint das Programm für den Beobachter nichts zu tun.

Erst jetzt (Zeile 340) wird die erste Seite auch angezeigt. Durch Warteschleifen getrennt wird nun ständig gewechselt.

Sie sehen also, in welcher Weise man mit zwei Graphikseiten arbeiten kann. Etwas weiter unten werden wir noch einen Befehl kennenlernen, mit dem wir diese beiden Graphikseiten noch logisch verknüpfen können. Hierdurch bieten sich Ihnen eine ganze Reihe von Möglichkeiten, die wir dort ebenfalls anschneiden werden.

zu m:

Mit m wählen Sie die Betriebsart Ihrer Graphikerweiterung. Für m sind Werte von 0-8 erlaubt. Für jeden Wert existiert eine bestimmte Graphikkonfiguration:

m=0: Textmodus bzw. LGR werden angezeigt und auch befehligt
m=1: HGR wird angezeigt und befehligt
m=2: MC wird angezeigt und befehligt
m=3: HGR wird angezeigt, LGR jedoch gefehligt, d.h. die Graphikbefehle beziehen sich auf die unsichtbare Low-Graphik
m=4: MC wird angezeigt, jedoch LGR befehligt
m=5: LGR und normaler Text werden angezeigt, die Befehle gelten aber für die hochauflösende Graphik
m=6: LGR bzw. Text werden angezeigt, MC jedoch befehligt
m=7: Gemischte Text- und Graphikanzeige, LGR wird befehligt. Im Anschluß wird die Angabe der Fensterposition verlangt
m=8: Gemischte Text- und Graphikanzeige, HGR wird befehligt; Angabe der Fensterposition

Sie können, während Sie z.B. Ihren Text oder Ihr LGR-Bild auf dem Bildschirm anzeigen, eine hochauflösende Graphikseite befehligen, d.h. alle Befehle gelten für HGR (Modus 5). Das ganze funktioniert ähnlich, wie bei der Behandlung der beiden Graphikseiten (s. obiges Programm).

Eine Anmerkung zu m=7/8:

Erinnern Sie sich an das erste Programm dieses Abschnittes? Dort wurde der Modus m=7 angewandt. Diese beiden Modi eröffnen Ihnen eine völlig neue Dimension: Text und Graphik werden gleichzeitig auf dem Bildschirm dargestellt. Sie haben die Möglichkeit, durch die Angabe zweier weiterer Parameter die Ober- und Unterkante des Textfensters anzugeben. Die entsprechenden Befehle lauten dann:

```
GMODE 0,7,100,150   oder
GMODE 0,8,150,100
```

Im ersten Fall gelten alle Befehle für LGR, ein kleines Textfenster unterbricht das eigentliche Graphikfenster.
Im zweiten Fall gelten alle Befehle für HGR, da aber der erste anschließende Wert größer ist als der zweite, erscheint nun ein kleines Graphikfenster mitten im Text. Einschränkend sind zwei Dinge zu bemerken:

- Wird der Fensterrand mitten in eine Textzeile verlegt, so erscheinen ab und zu merkwürdige Zeichen, die auf die hardwaremäßige Bearbeitung des Bildes durch den VIC zurückzuführen sind. Gleichfalls werden Sprites nicht abgeschnitten, sondern "zerstückelt".

- Gleichzeitige Text- und Graphikanzeige schließen die Anzeige von 16 Sprites aus und umgekehrt. Jeweils der zuletzt gewählte Modus hat Priorität.

Ich brauche Ihnen nicht zu sagen, welche Möglichkeiten Ihnen mit dem GMODE-Befehl offenstehen. Sie können mit den verschiedenen Graphikarten je nach Belieben wie ein Profi jonglieren. Noch etwas zur Syntax: Sie können den Befehl auf mehrerlei Art benutzen, um Ihnen die Arbeit zu erleichtern.

- GMODE s
- GMODE ,m
- GMODE s,m
- GMODE ,m,or,ur (nur bei m=7/8)
- GMODE s,m,or,ur (nur bei m=7/8)

Noch eine kleine Anmerkung: beim Drücken der beiden Tasten (run/stop) und (restore) wird grundsätzlich der Befehl GMODE ,0 ausgeführt, so daß Sie stets wieder in den Textmodus zurückgelangen. Desgleichen passiert, wenn irgendwann einmal ein Fehler auftritt, damit Sie immer auf dem Laufenden sind. Wird ein Programm ordnungsgemäß beendet (durch END oder durch Erreichen des Programmendes), so wird ebenfalls zurückgeschaltet. Nichts dergleichen geschieht, wenn sie Ihr Programm durch den Befehl STOP oder nur durch die Taste (run/stop) unterbrechen.

Aber probieren geht über studieren! Versuchen Sie es doch einmal - Sie werden staunen!

```
100 REM ********************************************
110 REM **                                        **
120 REM **  TIEF VERSTECKT, HALB VERDECKT, ...    **
130 REM **                                        **
140 REM ********************************************
150 REM
160 GMODE 0,5 : GCLEAR  :REM TEXT EIN, GRAPHIK BEFEHLIGEN
170 PRINT CHR$(147)     :REM TEXTBILDSCHIRM LOESCHEN
180 FOR X=1 TO 1000
190 PLOT ,320*RND(1),200*RND(1) :REM ZUFALLSPUNKTE
200 PRINT CHR$(19);X            :REM LAUFENDER ZAEHLER
210 NEXT X
220 GMODE ,1            :REM GRAPHIK EINSCHALTEN
230 WAIT 198,255 : LIST :REM AUF TASTE WARTEN + LIST
```

Da wir gerade dabei sind, wollen wir uns auch noch gleich einem anderen Befehl widmen, den wir ebenfalls ständig verwenden, ohne ihn vorgestellt zu haben. Das wollen wir hier gleich nachholen:

GCLEAR

Befehl:	GCLEAR (m)(,x1,y1 TO x2,y2)
Beispiel:	GCLEAR
	GCLEAR 255
	GCLEAR 4,30,40 TO 50,70
Parameter:	m : Löschmaske
	x1: x-Koordinate Fenster oben links
	y1: y-Koordinate Fenster oben links
	x2: x-Koordinate Fenster unten rechts
	y2: y-Koordinate Fenster unten rechts
Funktion:	Löschen des Graphikbildschirms bzw. eines Bildschirmfensters

Dieser Befehl dürfte Ihnen nun schon - ohne Parameter jedenfalls - vertraut vorkommen. Er löscht, wie Sie vielleicht schon gemerkt haben, die aktuelle HGR- oder MC-Bildseite. Gleichzeitig erhalten alle Punkte die aktuelle Punktcolor (s. PCOL=), es wird also ein SCOL=-Befehl ausgeführt (s.u.). In LGR verwenden Sie bitte PRINT CHR$(147) zum Löschen. Auch im LGR-Betrieb wird durch diesen Befehl die Graphikseite gelöscht!

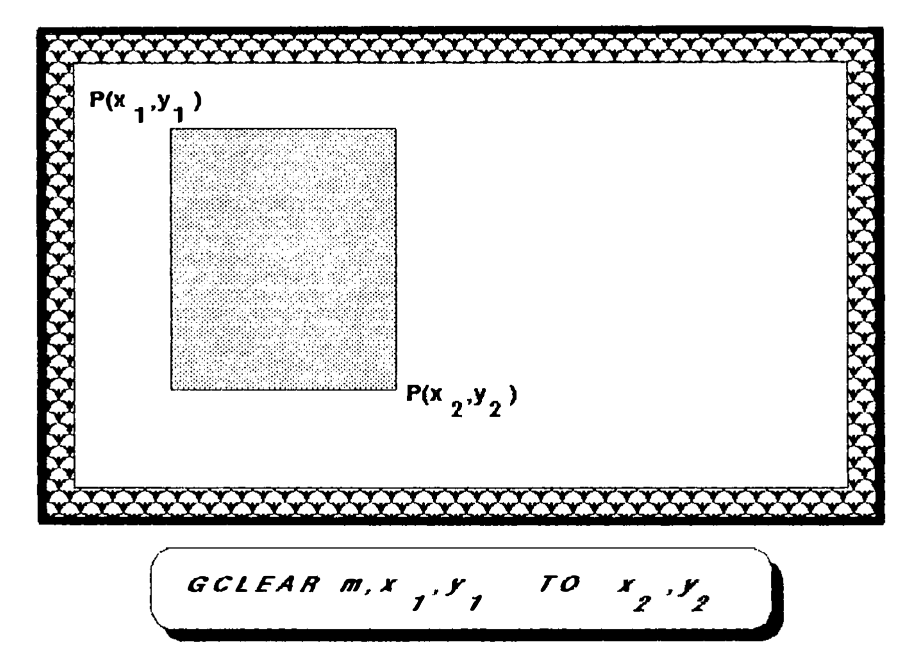
Dieser Befehl sollte vor jeder neuen Bearbeitung einer Graphikseite gegeben werden.

Der erste Parameter des GCLEAR-Befehls, den Sie bei Bedarf zusätzlich angeben können, bestimmt die sogenannte Lösch-Maske, d.h. der Wert des Bytes, der beim Löschen des Graphikbildes in jedes Graphikspeicherbyte geladen wird. Wenn Sie den hier dezimal angegebenen Wert binär umrechnen, so erhalten Sie ein Muster von Einsen und Nullen, das gesetzten bzw. nicht gesetzten Punkten entspricht. In Multicolor bestimmt dieses Byte gleichzeitig ein Farbmuster, mit dem gelöscht wird.

Zusätzlich zu diesem einen Parameter können Sie noch die Eckkoordinaten (x1,y1/x2,y2) eines Graphikfensters festlegen. Haben Sie dies getan, so wird nur dieses Fenster mit der Lösch-Maske (falls angegeben) gefüllt. Wie Sie diesen Fensterbefehl anzuwenden haben, erfahren Sie weiter unten beim GCOMB-Befehl.

Alles in allem haben Sie die folgenden Syntax-Möglichkeiten:

```
GCLEAR
GCLEAR m
GCLEAR  ,x1,y1 TO x2,y2
GCLEAR m,x1,y1 TO x2,y2
```



GCLEAR m,x1,y1 TO x2,y2

2.5.5 Super-Snake - unser erstes Spiel

Spiele sind oft etwas faszinierendes: Bunte Bälle flitzen über den Bildschirm, Flugzeuge, Raketen oder Männchen mühen sich ab, zu landen, zu treffen oder Treppen zu erklimmen. Es macht oft Spaß, sich stundenlang mit solchen Produkten der Phantasie irgendeines Programmierers zu beschäftigen. Doch Spiele selbst zu programmieren ist gar nicht so schwer, wie man vielleicht denkt. Man muß nur wissen, wie es ungefähr geht und schon wird man selbst zum "Spieleautor". Das Schöne dabei ist, man kann selbst bestimmen, wie das eigene Spiel aussehen soll. Wenn es irgendwann einmal langweilig wird, ändert man selbst etwas an seinem Programm und schon kommt vielleicht ein ganz neuer Witz ins Spiel.

Nun, wir werden hier bestimmt keine Super-Spiele vorstellen, mit denen man sich durch 123 Ebenen in atemberaubender Geschwindigkeit schlagen muß. Solche Spiele sind meist in Assembler geschrieben, und ein Buch würde nicht ausreichen, ein vollständiges Listing abzu-

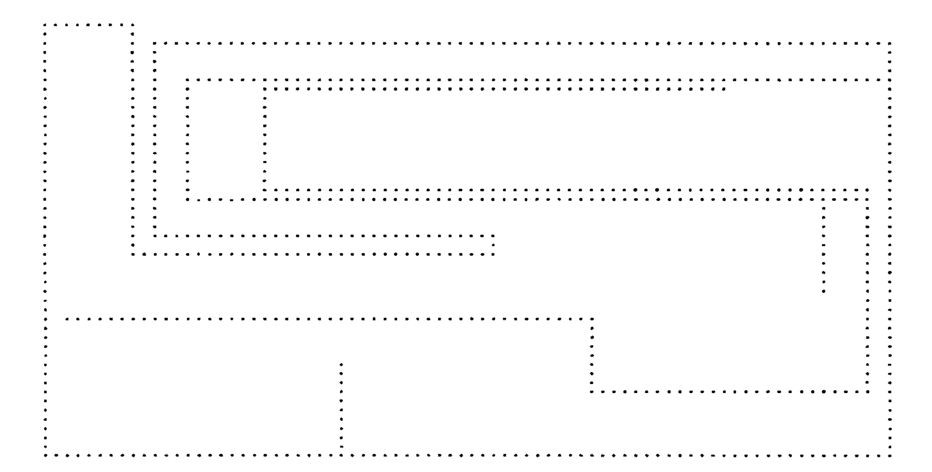
drucken. Die Entwicklungszeit solcher Spiele beträgt manchmal Monate intensiver Arbeit.

Aber man kann auch mit kleinen Mitteln interessante Effekte und Spielideen entwickeln. Das einzige Handwerkszeug hierzu ist Ihr Computer, die Supergraphik und nicht zuletzt Ihre Phantasie.

Mit dem folgenden - wirklich kleinen - Spiel wollen wir natürlich - wie immer - weiter dazulernen. Doch schauen Sie sich das Programm doch erst einmal an:

```
100 REM *******************
110 REM **               **
120 REM **  SUPER-SNAKE  **
130 REM **               **
140 REM *******************
150 REM
160 GMODE 0,1 : GCLEAR
170 COLOR=  2,10  :REM RAHMEN DUNKEL-, HINTERGRUND HELLROT
180 G=4           :REM GESCHWINDIGKEIT
190 REM
200 X=-G : Y=0    :REM STARTRICHTUNG: LINKS
210 A=160 : B=100 :REM STARTKOORDINATEN
220 GET A$        :REM TASTE HOLEN
230 IF A$="I" THENX= 0 : Y=-G :REM HOCH
240 IF A$="M" THENX= 0 : Y= G :REM RUNTER
250 IF A$="J" THENX=-G : Y= 0 :REM LINKS
260 IF A$="K" THENX= G : Y= 0 :REM RECHTS
270 A=A+X : B=B+Y :REM WEITERBEWEGEN
280 IF A<0 OR A>319 OR B<0 OR B>199 THEN 400
290 PLOT T TEST,0,A,B :REM TESTEN, OB BEREITS PUNKT GESETZT + PUNKT SETZ
EN
300 IF TEST <> 0 THEN 400
310 GOTO 220          :REM WEITER
320 REM
330 REM
340 REM **********************
350 REM **                  **
360 REM **  TREFFERROUTINE  **
370 REM **                  **
380 REM **********************
390 REM
400 FOR C=1 TO 10
410 FOR D=0 TO 15
420 COLOR= D,1 :REM RAHMEN BLINKEN LASSEN
430 NEXT D
440 NEXT C
450 A=INT(RND(1)*319/G)*G
```

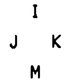
```
460 B=INT(RND(1)*199/G)*G
470 COLOR= 2,10
480 Z=Z+1    :REM ZAEHLER
490 IF Z=12 THEN 590
500 GOTO 220
510 REM
520 REM
530 REM ************
540 REM **        **
550 REM **  ENDE  **
560 REM **        **
570 REM ************
580 REM
590 GMODE 0,0        :REM GRAPHIK AUS
600 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
610 POS= 12,10       :REM CURSOR POSITIONIEREN
620 PRINT "KEINE PANIK !!!"
630 POKE 198,0       :REM TASTEN LOESCHEN
640 WAIT 198,255     :REM AUF TASTE WARTEN
650 COLOR= 0,0       :REM FARBEN RUECKSETZEN
```



Das vorstehende Spiel funktioniert folgendermaßen: Auf dem Graphikbildschirm bewegt sich ununterbrochen eine immer länger werdende Schlange. Sie haben nun die Aufgabe, mittels der Tasten

I für hoch
M für runter
J für links
K für rechts

entsprechend der Tastenanordnung:



auf der Schreibmaschinentastatur die Bewegungsrichtung dieser Schlange so zu steuern, daß sie sich weder aus dem Bildschirmfenster hinaus bewegt, noch gegen einen Teil Ihres Schwanzes stößt. In beiden Fällen bekommen Sie Punktabzug. Anschließend erscheint die Schlange wieder irgendwo auf dem Bildschirm. Bei 12 solchen Karambolagen ist das Spiel zu Ende. Sie haben natürlich die Möglichkeit, das Spiel nach Ihren Vorstellungen auszufeilen und zu verändern. Sie können es z.B. für zwei gegeneinander spielende Spieler umschreiben, Punkte zählen, Zeit messen, statt der Tastatursteuerung die Joysticks zu Hilfe nehmen und so weiter und so fort. Das Spiel ist extra so eingerichtet, daß Sie Änderungen gut und bequem vornehmen können.

Die Geschwindigkeit der Schlange beispielsweise können Sie in Zeile 180 leicht verändern (Die Variable G enthält die Geschwindigkeit = Schrittweite pro Durchlauf). In den Zeilen 230-260 werden die vier Tasten geprüft und dementsprechend die Richtung geändert. Zeile 270 dann errechnet die neuen Koordinaten. Die Speicher X und Y bestimmen jeweils die Anzahl der Schritte pro Durchlauf in x- bzw. y-Richtung. A und B enthalten die aktuellen Koordinaten des zu zeichnenden Punktes, die in Zeile 280 daraufhin geprüft werden, ob sie noch innerhalb des Graphikfensters liegen.

Dieses Beispiel enthält gleich zwei bisher noch unbekannte Befehle. Zum einen taucht in dem vorgestellten Programm mehrere Male der Befehl COLOR= auf. Mit ihm können Sie Rahmen und Hintergrundfarbe des aktuellen Bildschirmfensters festlegen:

COLOR=

Befehl:	COLOR= r,h	
Beispiel:	COLOR= 2,6	
Parameter:	r:	Rahmenfarbe
	h:	Hintergrundfarbe
Funktion:	Festlegen der Rahmen- und Hintergrundfarbe	

Mit h wird die Hintergrundfarbe des jeweiligen Graphikfensters festgelegt. Sie kann für LGR (=Text), Seite 1 und Seite 2 (HGR/MC) jeweils verschiedene Werte annehmen. Beim Umschalten auf einen anderen Graphikmodus wird Sie also mit berücksichtigt. Die Rahmenfarbe r bleibt stets erhalten. Wir werden auf diesen Befehl auch im Kapitel "Farbe" noch einmal zurkückkommen.

Besonders interessant für uns ist neben dem bereits Gesagten die Zeile 290. Hier taucht wiederum eine Besonderheit des PLOT-Befehls auf, die wir bisher noch nicht kennengelernt haben. Sie erinnern sich vielleicht an die erweiterte Syntax der Zeichenbefehle. Dort hatten wir uns bisher nur mit dem B-Zusatz beschäftigt (z.B. PLOT B m...). Hier haben wir es nun mit einem weiteren Zusatz zu tun:

T var - Punkte zählen:

Hier ist eine weitere Außergewöhnlichkeit Ihrer Graphikerweiterung: Der Test-Befehl. Er kann zusätzlich zu allen anderen Syntaxerweiterungen hinzugefügt werden, wobei allerdings die Reihenfolge beachtet werden muß, doch darauf kommen wir noch zu sprechen.

Mit diesem Befehl können Sie die bereits gesetzten Punkte zählen, die von Ihrer Figur angesprochen werden; sei es, Sie löschen, setzen, invertieren, punktieren (wobei die Punktierung ein Wechsel von Löschen und Setzen ist, also voll mitgezählt wird) oder bewegen nur den Graphikcursor. Letzteres ist besonders interessant, da Sie so lediglich zählen, ohne auf dem Bildschirm etwas zu verändern.

Die sich beim Zählen ergebende Zahl steht dann als Rückmeldung in der angegebenen Variable (Zahlen über 32768 sind ähnlich der FRE(0)-Meldung negativ). Anmerkung: In Zusammenhang mit DRAW z.B. könnten Sie ein mit DRAW gezeichnetes Objekt, das druch stetes Löschen und verschoben wieder Setzen über den Bildschirm bewegt wird, auf Kollisionen mit Teilen des Bildschirms (ähnlich Sprites) überprüfen.

Wird der T-Zusatz gemeinsam mit dem PLOT-Befehl verwandt, wie dies im vorstehenden Programm der Fall ist, so dient er einfach dazu zu testen, ob sich an der angesprochenden Stelle bereits ein Punkt befindet. In unserer Anwendung wird dann gleichzeitig in einem Befehl genau an diese Stelle dann (also nachdem getestet wurde) ein Punkt gesetzt, da wir im Modus 0 zeichnen. In der Variable TEST (der Name ist natürlich frei wählbar) steht dann entweder eine "0", wenn an der PLOT-Stelle vorher kein Punkt stand oder eine "1", wenn an der Stelle tatsächlich bereits früher schon ein Punkt gesetzt war.

Das folgende Programm sollte Ihnen den Zusammenhang noch etwas verdeutlichen:

```
100 GMODE 0,8,161,250 : GCLEAR : SCOL= 0,0
120 POS= 0,20
130 GOSUB 180      :REM TESTEN, OB PUNKT GESETZT
140 PLOT ,100,100 :REM PUNKT SETZEN
150 GOSUB 180      :REM TESTEN, OB PUNKT GESETZT
160 END
170 REM
180 PLOT T RUECK,4,100,100     :REM TESTEN OHNE ZEICHNEN
190 IF RUECK=1 THEN PRINT "PUNKT IST GESETZT!" : GOTO 210
200 PRINT "PUNKT IST GELOESCHT!"
210 POKE 198,0 : WAIT 198,255  :REM AUF TASTE WARTEN
220 RETURN
```

Wie gesagt, können die zwei bisherigen Zusätze (insgesamt sind es drei, den letzten werden wir im Farbkapitel kennenlernen) sowohl einzeln, als auch im Verbund auftauchen. Hier sollen Ihnen anhand des PLOT-Befehls zum Zeichnen eines Punktes die bisherigen Möglichkeiten der Syntaxerweiterung dargelegt werden:

```
PLOT              (zm),x,y
PLOT B m,         (zm),x,y
PLOT      T var,  (zm),x,y
PLOT B m, T var,  (zm),x,y
```

Die Leerzeichen zwischen den einzelnen Zeichen stehen hier natürlich nur der Übersichtlichkeit wegen und können weggelassen werden. zm steht in Klammern, da bei Weglassen automatisch der Wert 0 eingesetzt wird.

2.6 Linien, der gemeinsame Nenner der Natur

So wie wir vorhin den Punkt als Ursprung aller Graphik erkannt haben, so werden wir einsehen müssen, daß in der praktischen Arbeit kaum noch mit einzelnen Punkten gearbeitet wird. Hier stellt die Linie das Hauptmotiv dar, aus dem alle anderen Figuren konstruiert werden. Der Punkt wird dann nur noch zu einem Sonderfall der Linie degradiert, einer Linie mit gleichem Start- wie Endpunkt. Der Punkt als der Baustein der Linie verschwindet völlig aus dem Vorstellungsbereich der Computergraphiker. Auch Kreise, Ellipsen oder andere scheinbar runde Gebilde werden durch eine Vielzahl von Linien angenähert. Hierdurch werden sie viel leichter berechenbar. Eine Fläche wird dann zu einem allgemeinen Polygon, also einem allgemeinen Vieleck. Diese Vereinfachung erlangt in allen Bereichen der Computergraphik zentrale Bedeutung. Die Linie wird quasi zum gemeinsamen Nenner der Graphik.

Mit der immensen Wichtigkeit des graphischen Elements Linie, steigt auch die Wichtigkeit einer optimalen, schnellen und komfortablen Liniendarstellung auf dem Bildschirm oder anderen Ausgabegeräten. Aus diesem Grunde wurde der Supergraphik auch eine besonders schnelle Linienberechnung mitgegeben. Wenn Sie zufällig im Besitz einer anderen Graphik- oder BASIC-Erweiterung sind, so scheuen Sie sich nicht, einmal einen Geschwindigkeitsvergleich anzustellen. Sie werden erstaunt sein, wie schnell der Supergraphikbefehl trotz allen Komforts ist.

2.6.1 PLOT ... TO ...

PLOT ... TO ...

Befehl:	PLOT zm,x1,y1 TO x2,y2 (TO x3,y3 TO ...)	
Beispiel:	PLOT 0,100,199 TO 0,20	
Parameter:	zm:	Zeichenmodus
	x1:	x-Startkoordinate
	y1:	y-Startkoordinate
	x2:	x-Zielkoordinate
	y2:	y-Zielkoordinate
Funktion:	Setzen (Loeschen etc.) einer oder mehrerer Linien	

Da ist er nun, der Befehl, der uns eine ganz neue Welt eröffnet. Das langsame Herumschleichen mit dem einfachen PLOT-Befehl hat nun ein Ende. Nicht nur die Geschwindigkeit wird zunehmen, die Möglichkeiten der Graphikgestaltung werden um ein Vielfaches erweitert. Doch schauen Sie sich erst einmal die Anwendung dieses Befehls an:

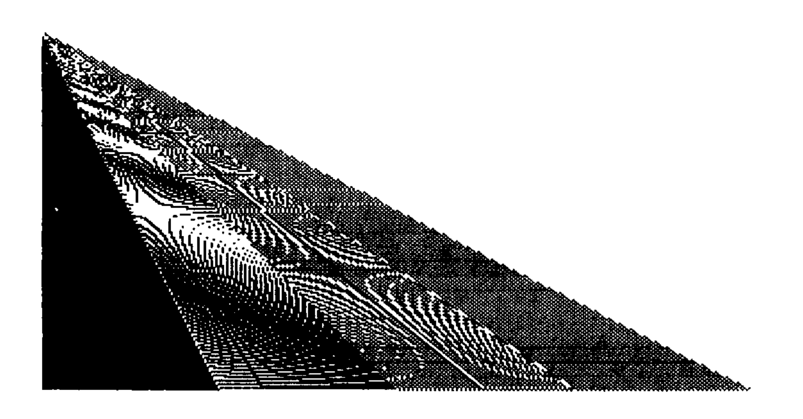
```
120 REM **  DAS HAUS VOM NIKOLAUS  **
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 A= 5 : B=10 : C=15
180 D= 2 : E=10 : F=18
190 S= 2                      :REM GESCHWINDIGKEIT
200 REM
210 FOR X=0 TO 230/S
220   PLOT X/8,A,F TO A,E TO B,D TO C,E TO A,E TO C,F TO A,F TO C,E TO C
,F
230   A=A+S : B=B+S : C=C+S
240   D=D+S : E=E+S : F=F+S
250   PLOT  ,A,F TO A,E TO B,D TO C,E TO A,E TO C,F TO A,F TO C,E TO C,F
260 NEXT X
270 WAIT 198,255
```



In diesem Programm wird durch einen einzigen Befehl ein ganzes Haus gezeichnet (Zeile 250) bzw. gelöscht (Zeile 220). Dabei werden stets die Punkte mit den zwischen den trennenden "TO" stehenden

Koordinaten miteinander verbunden. Die Länge eines PLOT ... TO ...-Befehls ist dabei theoretisch beliebig und nur durch die Länge des Eingabepuffers bzw. einer BASIC-Zeile begrenzt. Sie können den Befehl - und das wird wohl am häufigsten vorkommen - natürlich auch nur zum Zeichnen einer einzigen Linie verwenden. Dabei beschränken Sie sich dann auf die Angabe zweier Koordinatenpaare und des Zeichenmodus.

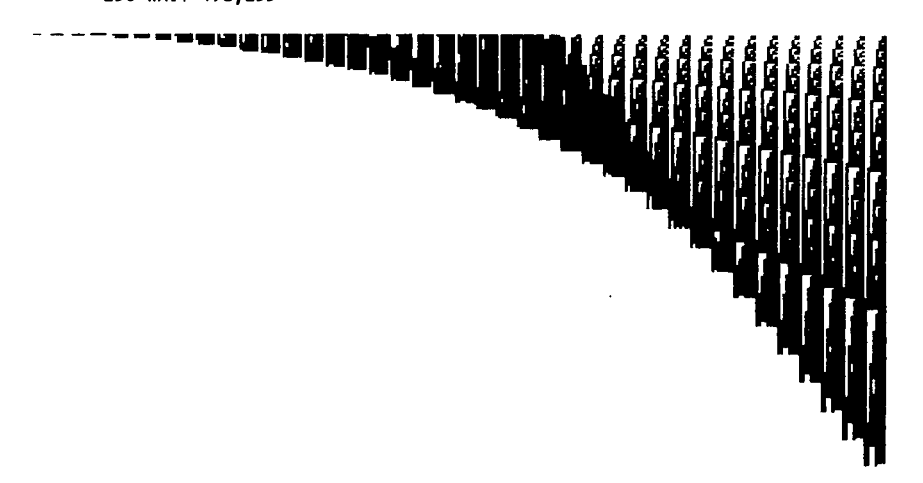
Wie Sie sehen und wie auch unter PLOT bereits angekündigt, können Sie auch hier den Zeichenmodus angeben. Dabei bezieht sich alles unter PLOT Gesagte hier dann auf die gezeichnete, gelöschte oder invertierte Linie. Interessante Effekte lernen Sie dabei im nächsten Paragraphen kennen. Sinn bekommt in diesem Zusammenhang nun auch der Punktier-Modus zm=3, der beim einfachen PLOT-Befehl noch nicht vorgeführt werden konnte. Hier ein Beispiel:



```
100 REM *******************
110 REM **               **
120 REM **  ZEICHENMODI  **
130 REM **               **
140 REM *******************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR X=0 TO 160
180   PLOT ,0,0 TO X,199
190 NEXT X
200 REM
210 FOR X=160 TO 319
220   PLOT 3,0,0 TO X,199
230 NEXT X
240 REM
250 FOR X=80 TO 240
260   PLOT 2,0,0 TO X,199
270 NEXT X
280 WAIT 198,255
```

Vergessen Sie bitte nicht, daß die erweiterte Syntax auch auf diesen Befehl - wie auch auf jeden anderen der zukünftig vorgestellten Befehle - angewendet werden kann! So könnte z.B eine kleine Anwendung des B-Zusatzes aussehen:

```
100 REM ************
110 REM **        **
120 REM **  DEMO  **
130 REM **        **
140 REM ************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR X=1 TO 200
180   PLOT B X,,0,X-1 TO 319,X-1
190 NEXT X
200 FOR X=1 TO 5000 : NEXT X
210 GCLEAR
220 FOR X=1 TO 200
230   PLOT B X,,X-1,0 TO 319,X-1
240 NEXT X
250 WAIT 198,255
```



Der T-Zusatz, den wir ebenfalls unter PLOT bei unserem kleinen Super-Snake-Spiel durchgesprochen haben, erlangt beim Zeichnen von Linien eine erweiterte Bedeutung. Es wird nicht nicht getestet, ob die Linie einen bereits gezeichneten Punkt berührt, es werden vielmehr alle die Punkte gezählt, die die Linie überzeichnet. In der dem T-Zusatz folgenden Variable (in unserem Super-Snake-Spiel TEST genannt) wird dann die Anzahl dieser Punkte zurückgegeben.

PLOT TO ...

Befehl:	PLOT zm, TO x2,y2 (TO x3,y3 TO ...)
Beispiel:	PLOT 2, TO 100,199
Parameter:	zm: Zeichenmodus
	x2: x-Zielkoordinate
	y2: y-Zielkoordinate
Funktion:	Setzen (Loeschen etc.) einer Linie vom Graphikcursor

Sie sollten wissen, daß sich Ihre Supergraphik jeweils den zuletzt gezeichneten, gelöschten, invertierten, kurz jeden zuletzt angesprochenen Punkt als unsichtbaren Graphikcursor merkt. D.h. wenn Sie mit dem PLOT-Befehl einen Punkt oder eine Linie zeichnen, so wird dieser Punkt bzw. der letzte Punkt der Linie gespeichert. Es gibt nun Befehle, die von diesem Graphikcursor, also von diesem letzten Punkt Gebrauch machen. Einer dieser Befehle ist PLOT TO ... Bei diesem Kommando brauchen Sie neben dem Zeichenmodus lediglich eine weitere Koordinate anzugeben, um eine Linie von dem Punkt, an dem der unsichtbare Cursor steht, bis zu dem Punkt mit den von Ihnen festgelegten Koordinaten zu zeichnen. Selbstverständlich können Sie wie beim PLOT ... TO ...-Befehl soviele TO ... anhängen, wie es Ihnen beliebt.

Eine Sache sollten Sie beachten: Wie Sie oben sehen, müssen Sie zwischen PLOT und TO den Zeichenmodus angeben. Dabei ist das Komma hinter dieser Zahl unbedingt notwendig. Also z.B.: PLOT 2, TO 100,20

Der unsichtbare Graphikcursor gibt Ihnen mannigfaltige Möglichkeiten in die Hand. Zum einen können Sie Linienzüge mit PLOT TO ... weiterführen, wie wir das im nächsten Programm sehen werden. Zum anderen können Sie im Zeichenmodus 4 (s. PLOT) Punkte oder Linien (später auch Kreise etc.) zeichnen, ohne daß sich etwas auf dem Bildschirm verändert. Im Zeichenmodus 4 wird dann nur der Graphikcursor bewegt, denn die Koordinaten des letzten Punktes werden natürlich auch bei diesem Befehl gespeichert.

Im nächsten Programm stellen wir Ihnen eine kleine Anwendung vor, die sich des Graphikcursors bedient:

```
100 REM **********************
110 REM **                  **
120 REM **  FUNKTIONSKISTE  **
130 REM **                  **
140 REM **********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 DEF FN F(X)= 50*SIN(LOG(X)*5)+50+S*5 :REM FUNKTION DEFINIEREN
180 FOR S=21 TO 1 STEP -4     :REM STEP-RATE VERAENDERN
190   PLOT 4,1,FN F(1)        :REM GRAPHIKCURSOR FUER X=1 POSITIONIEREN
200   FOR X=2 TO 319 STEP S
210     PLOT 0, TO X,FN F(X) :REM LINIE ZEICHNEN
220   NEXT X
230 NEXT S
240 WAIT 198,255
```

Wenn Sie sich etwas in BASIC auskennen, dann wird Ihnen bekannt sein, daß in Zeile 170 eine Funktion definiert wird, die im späteren Verlauf des Programms jedesmal für den Ausdruck FN F(X) eingesetzt wird (Zeilen 190 und 210). Dabei wird stets der Wert in Klammern für das X in der Funktion eingesetzt. Selbstverständlich behalten alle anderen Variablen, die in der Funktion vorkommen (hier S) ihre Gültigkeit.

In Zeile 190 wird als Startpunkt nur der Cursor bewegt. Dies ist notwendig, damit die später vom Graphikcursor aus bewegte Linie (Zeile 210) vom richtigen Punkt ausgeht. Wie Sie sehen, bestimmt die äußere der beiden FOR...NEXT-Schleifen den STEP-Wert der inneren. Beim ersten äußeren Schleifendurchlauf wird nur an jeder 21. Stelle ein Punkt gezeichnet. Trotzdem wird jeder berechnete Punkt der Kurve mit dem vorherigen verbunden, es entsteht also ein - wenn auch sehr zackiger - kontinuierlicher Verlauf. Daß PLOT TO auch nicht versagt, wenn die beiden zu verbindenden Punkte direkt nebeneinander liegen, zeigt dann der letzte Schleifendurchlauf mit S=1.

Ersetzen Sie doch einmal Zeile 210 durch:

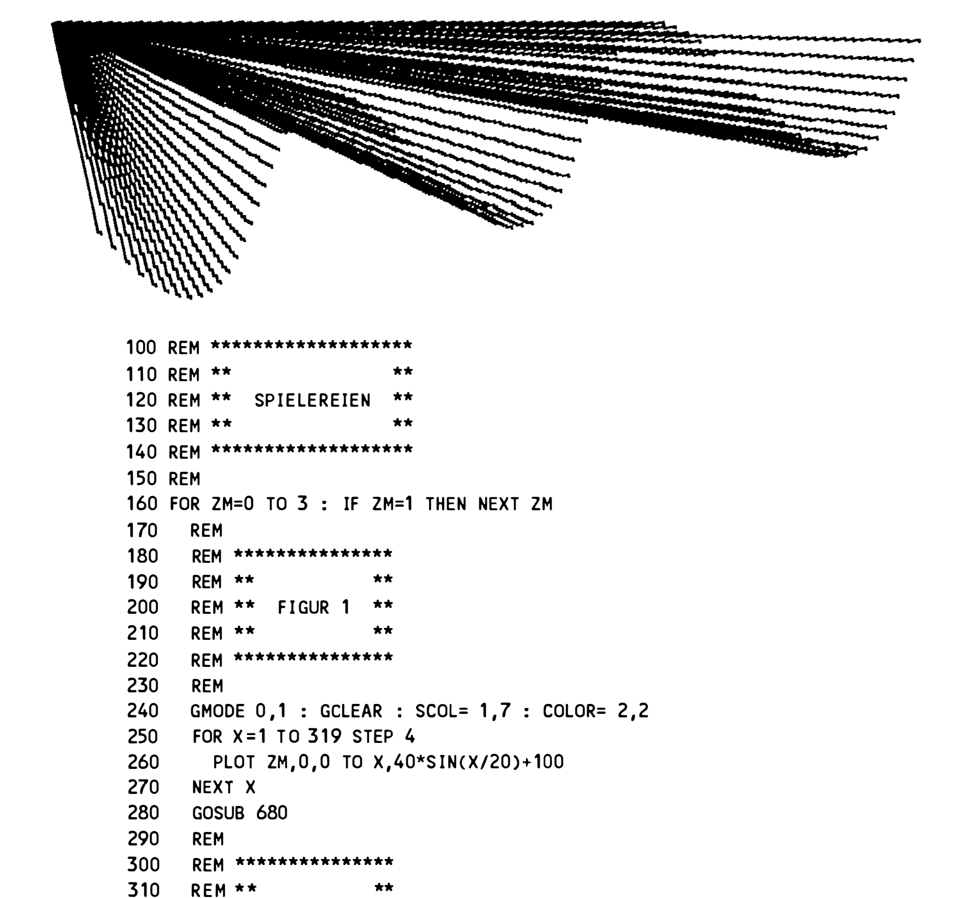
```
210 PLOT B 23,0, TO X,FN F(X) :REM LINIE ZEICHNEN
```

oder durch:

```
210 PLOT B S ,0, TO X,FN F(X) :REM LINIE ZEICHNEN
```

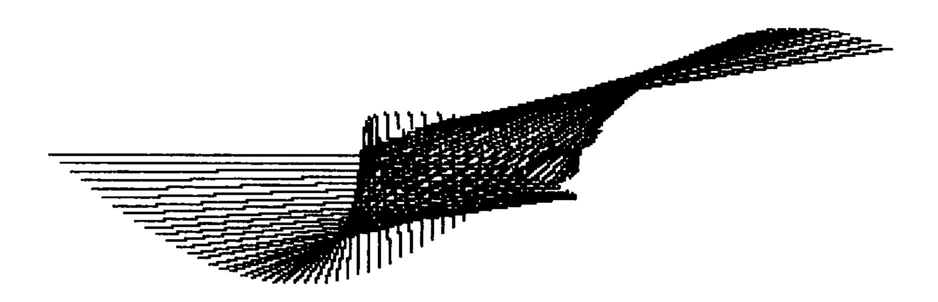
2.6.2 Kleine Spielereien

Mit den folgenden Ausführungen wollen wir Ihnen demonstrieren, wie mannigfaltig die Möglichkeiten der Graphikgestaltung bereits in diesem Stadium sind. Vornehmlich der Linienbefehl wird hier eine Rolle spielen. Sie brauchen nicht unbedingt jedes Detail eines Programms zu verstehen, sollten jedoch überall einmal Änderungen vornehmen, um den resultierenden Effekt zu beobachten. Schon kleine Änderungen können das ganze Bild föllig verändern. Mit der Zeit bekommen Sie sicher Übung (besonders, wenn Sie sich etwas in der Mathematik auskennen und wenn Sie unsere Ausführungen im PLOT-Kapitel beherzigen, die uns etwas über Streckung, Verschiebung etc. von Sinus-Funktionen sagten). Doch fangen wir gleich einmal an:



```
100 REM *******************
110 REM **               **
120 REM **  SPIELEREIEN  **
130 REM **               **
140 REM *******************
150 REM
160 FOR ZM=0 TO 3 : IF ZM=1 THEN NEXT ZM
170   REM
180   REM ***************
190   REM **           **
200   REM **  FIGUR 1  **
210   REM **           **
220   REM ***************
230   REM
240   GMODE 0,1 : GCLEAR : SCOL= 1,7 : COLOR= 2,2
250   FOR X=1 TO 319 STEP 4
260     PLOT ZM,0,0 TO X,40*SIN(X/20)+100
270   NEXT X
280   GOSUB 680
290   REM
300   REM ***************
310   REM **           **
```

```
320   REM **  FIGUR 2  **
330   REM **           **
340   REM ***************
350   REM
360   GCLEAR
370   FOR X=1 TO 350 STEP 1
380     A=50*SIN(X/10)+160
390     B=60*SIN(X/20)+100
400     C=70*SIN(X/30)+160
410     D=80*SIN(X/40)+100
420     PLOT ZM,A,B TO C,D
430   NEXT X
440   GOSUB 680
450   REM
460   REM ***************
470   REM **           **
480   REM **  FIGUR 3  **
490   REM **           **
500   REM ***************
510   REM
520   GCLEAR
530   FOR X=1 TO 319 STEP 4
540     A=X
550     B=60*SIN(X/60)+100
560     C=40*SIN(X/25)+160
570     D=20*SIN(X/30)+100
580     PLOT ZM,A,B TO C,D
590   NEXT X
600   GOSUB 680
610   REM
620   NEXT ZM
630   END
640   REM
650   REM
660   REM ZEITSCHLEIFE:
670   REM
680   FOR T=1 TO 3000
690     GET A$ : IF A$="" THEN NEXT T
700   RETURN
```



Damit Sie vielleicht doch ein wenig mit den obigen Formeln anfangen können, hier einige Erläuterungen:

In der ersten Figur wurde jeder Punkt einer Sinuskurve mit dem Punkt (0,0) verbunden. Sinuskurven haben wir bereits unter PLOT kennengelernt.

Die zweite Figur setzt sich schon etwas komplizierter zusammen:
Nehmen wir an, Sie zeichnen in ein normales Kartesisches Koordinatensystem eine Sinuskurve. Normalerweise würden wir hierzu für jedes ganzzahlige x zwischen 0 und 319 einen y-Wert berechnen. Das tun wir auch. Wir sind aber nicht unbedingt gezwungen, diese x-Werte auch für die x-Koordinaten zu verwenden. Wenn wir nun die x-Koordinaten nicht (wie die x-Werte) von 0 bis 319 langsam aufsteigen lassen, sondern ebenfalls als Funktionswerte einer Sinuskurve (abhängig von den x-Werten) auffassen, so erhalten wir alles andere, nur keine Sinuskurve mehr. Diese Kurven heißen - wir haben sie bereits kennengelernt - Lissajouskurven. Wenn wir nun jeden Punkt einer Lissajouskurven mit einem korrespondierenden Punkt einer völlig anderen verbinden, so erhalten wir Figuren wie Figur 2.

In Figur 3 wurde nun jeder Punkt einer Sinuskurve mit einem korrespondierenden Punkt einer Lissajouskurve verbunden.

Um Ihnen zusätzlich noch den Effekt verschiedener Zeichenmodi zu demonstrieren, wurde um alle Figuren eine FOR...NEXT-Schleife gelegt, die den Zeichenmodus bestimmt.

Statt Sinus- oder Lissajouskurven können Sie natürlich jede andere Kurve verwenden. Sie müssen nur darauf achten, daß die Werte nicht außerhalb des Graphikfensters gelangen.

```
100 REM ************************
110 REM **                    **
120 REM **   SPIELEREIEN II   **
130 REM **                    **
140 REM **  NACH  IDEEN  VON  **
150 REM **   NORBERT TREITZ   **
160 REM **                    **
170 REM ************************
180 REM
190 PI=3.14159265
200 REM
210 REM ***************
220 REM **           **
230 REM **  FIGUR 1  **
240 REM **           **
250 REM ***************
260 REM
270 GMODE 0,1 : GCLEAR : SCOL= 1,7 : COLOR= 2,2
280 R=141
290 A=160 : B=100
300 FOR W=PI/4 TO 3.6 STEP 0.05
310   X=R*COS(W) : Y=R*SIN(W)
320   PLOT ,A+X,B-Y TO A-Y,B-X
330   PLOT ,A-Y,B-X TO A-X,B+Y
340        REM PLOT ,A-X,B+Y TO A+X,B-Y
350   PLOT ,A-X,B+Y TO A+Y,B+X
360   PLOT ,A+Y,B+X TO A+X,B-Y
370   R=R*0.94
380 NEXT W
390 GOSUB 680
400 REM
410 REM ***************
420 REM **           **
430 REM **  FIGUR 2  **
440 REM **           **
450 REM ***************
460 REM
470 FOR N=4 TO 100 STEP 5
480   GCLEAR
490   R1=99 : R2=50
500   DW=PI/N*2
510   FOR I=0 TO 4
520     R3=99-R1 : R4=99-R2
530     FOR W=DW TO 2*PI+DW*2 STEP DW
540       A=160+R3*COS(W)
550       B=100-R3*SIN(W)
560       PLOT ,A,B TO 160+R4*COS(W+DW),100-R4*SIN(W+DW)
570       PLOT ,A,B TO 160+R4*COS(W-DW),100-R4*SIN(W-DW)
580     NEXT W
590     R1=R1/2 : R2=R2/2
```

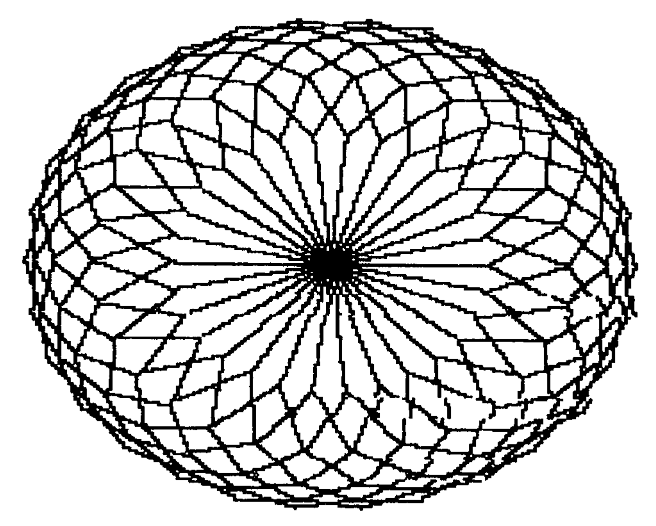
```
600   NEXT I
610   GOSUB 680
620 NEXT N
630 END
640 REM
650 REM
660 REM WARTESCHLEIFE:
670 REM
680 FOR T=1 TO 4000
690   GET A$ : IF A$="" THEN NEXT T
700 RETURN
```

Die erste dieser Figuren stellt ein sich drehendes Quadrat dar, das sich bis zur Unkenntlichkeit verkleinert. Grundlage dieses Programms ist die Kreisformel mit:

x=r*cos(w)+mx
y=r*sin(w)+my

mit
- x: x-Koordinate eines Kreispunktes
- y: y-Koordinate eines Kreispunktes
- w: Winkel zwischen y-Achse und Linie: Kreispunkt-Mittelpunkt
- r: Radius
- mx: x-Koordinate Kreismittelpunkt
- my: y-Koordinate Kreismittelpunkt

Die vier Eckpunkte des Quadrates werden als Punkte eines Kreises gesehen, dessen Mittelpunkt im Nullpunkt eines rechtwinkligen Koordinatensystems liegt. Im Laufe der Routine werden



a) der Radius verringert
b) die Eckpunkte des Quadrats um einen Winkel w um den Mittelpunkt des Kreises gedreht

Wie das ganze genau funktioniert, erfahren Sie im Grundlagenkapitel unter "Kreisberechnung" beim Thema "Achtelkreistechnik".

Die zweite Figur (oder die Figuren, denn es werden eine ganze Menge Figuren gezeichnet) zeigt einen hübschen Effekt, der ebenfalls auf der Kreisberechnung basiert (oder auf der Anwendung von Polarkoordinaten, je nachdem, was einem lieber ist). Verändern Sie auch hier ruhig ein paar Parameter, um zu sehen, was passiert.

2.6.3 Rechtecke - leicht gemacht

Wir wollen Rechtecke zeichnen. Nichts leichter als das, werden Sie sagen, wir haben ja einen Befehl zum Zeichnen von Linien. Vier Linien ergeben ein Rechteck und damit ist das Problem gelöst. Nun gut hier ist das BASIC-Programm:

```
100 REM *****************
110 REM **             **
120 REM **  RECHTECKE  **
130 REM **             **
140 REM *****************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR X=0 TO 150 STEP 7
180   X1=2*X  : Y1=X
190   X2=X+80 : Y2=X+50  :REM ECKPUNKTE FESTLEGEN
200   ZM=0               :REM ZEICHENMODUS
210   D =5               :REM RAHMENDICKE
220   GOSUB 330          :REM RAHMEN ZEICHNEN
230 NEXT X
240 WAIT 198,255
250 END
260 REM
270 REM RECHTECKE:
280 REM UEBERGABE: X1,Y1 - ECKPUNKT 1
290 REM            X2,Y2 - ECKPUNKT 2
300 REM            D     - DICKE
310 REM            ZM    - ZEICHENMOD.
320 REM
330 FOR T=0 TO D-1
340   XA=X1+T : YA=Y1+T
```

```
350    XB=X2-T : YB=Y2-T
360    PLOT ZM,XA,YA TO XA,YB
370    PLOT ZM,XA,YB TO XB,YB
380    PLOT ZM,XB,YB TO XB,YA
390    PLOT ZM,XB,YA TO XA,YA
400 NEXT T
410 RETURN
```

Wir haben es hier mit einem recht komfortablen Rechteckbefehl zu tun. Neben Zeichenmodus und Eckpunkten können Sie zusätzlich die Dicke des Rechteckrahmens festlegen. Gut und schön, wir haben verwirklicht, was wir uns in den Kopf gesetzt haben. Warum aber sollten wir uns jedesmal solche Umstände machen, wenn Supergraphik uns bereits einen Rechteckbefehl mit allen diesen Features bietet. Das obige Programm könnte auch so aussehen:

```
100 REM *****************
110 REM **             **
120 REM **  RECHTECKE  **
130 REM **             **
140 REM *****************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR X=0 TO 150 STEP 7
180   FRAME ,5,2*X,X TO X+80,X+50
190 NEXT X
200 WAIT 198,255
```

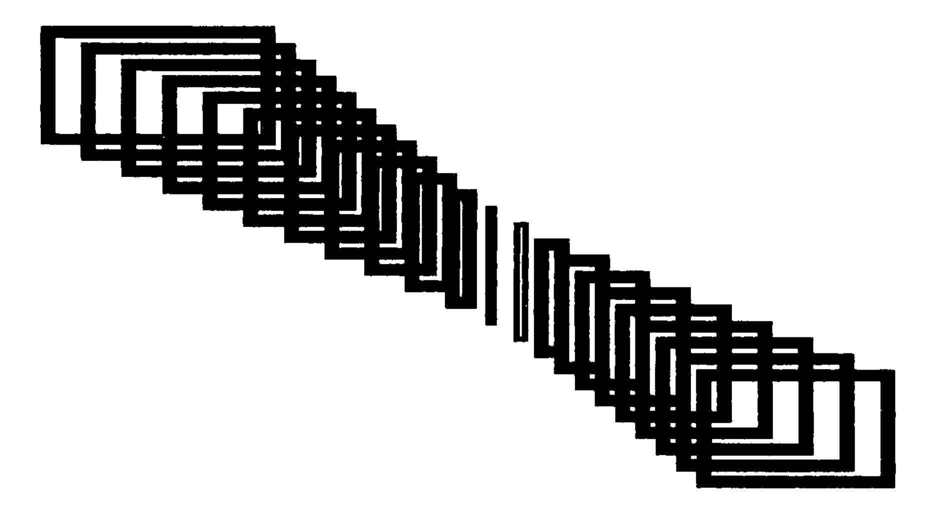
Ein bißchen kürzer, ein bißchen schneller, ein bißchen richtiger.

Kürzer? Das werden Sie wahrscheinlich bei ersten Augenschein erkennen. Schneller? Vergleichen Sie, der Unterschied ist deutlich. Richtiger? Ja, sehen Sie, wenn Sie im ersten Programm für das erste Koordinatenpaar kleinere Werte einsetzen als für das zweite, dann geht noch alles in Ordnung. Im Laufe der obigen FOR...NEXT-Schleife aber wird auch irgendwann das Gegenteil zutreffen. Schauen Sie sich doch dann einmal das Rechteck an.

Aber was denn? Wir haben unseren neuen Befehl ja noch gar nicht vorgestellt! Das ist sofort nachzuholen:

FRAME

Befehl:	FRAME zm,d,x1,y1 TO x2,y2
Beispiel:	FRAME 0,10,100,100 TO 200,150
Parameter:	zm: Zeichenmodus
	d: Rahmendicke
	x1: x-Eckkoordinate 1
	y1: y-Eckkoordinate 1
	x2: x-Eckkoordinate 2
	y2: y-Eckkoordinate 2
Funktion:	Setzen (Loeschen etc.) eines Rahmens definierter Dicke



Es wird ein Rahmen der Dicke d (Einheit 1 Punkt; für d=0 wird kein Rahmen gezeichnet) begrenzt von den angegebenen zwei gegenüberliegenden Eckpunkten gezeichnet. Wird d so groß, daß zwei Rahmeninnengrenzen aneinanderstoßen, entsteht eine zusammenhängende Fläche.

```
10 GMODE 0,1 : GCLEAR
20 FOR ZM=0 TO 3
30   FRAME ZM,50,0,0 TO 319,199
40 NEXT ZM
50 WAIT 198,255
```

Man staune, es wird ein Rahmen um das ganze Graphikfeld mit der Dicke von 50 Punkten gezogen, abgezogen, inverzogen, punzogen.

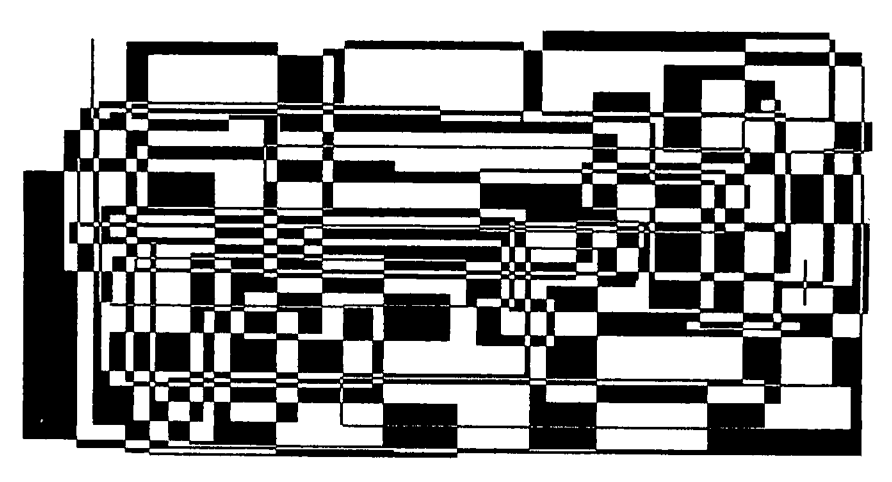
Kaum zu glauben, daß selbst bei diesem Befehl die Syntaxerweiterung, die wir unter PLOT und PLOT...TO... kennengelernt haben, einwandfrei funktioniert. Probieren Sie es aus!

Da wir gerade dabei sind, sollten wir uns auch gleich mit einen nahen Verwandten des FRAME-Befehls bekannt machen:

FILL

Befehl:	FILL zm,x1,y1 TO x2,y2	
Beispiel:	FILL 0,100,100 TO 200,150	
Parameter:	zm:	Zeichenmodus
	x1:	x-Eckkoordinate 1
	y1:	y-Eckkoordinate 1
	x2:	x-Eckkoordinate 2
	y2:	y-Eckkoordinate 2
Funktion:	Setzen (Loeschen etc.) eines Feldes	

Neben dem Rahmen gibt es auch die Möglichkeit, ein volles Feld zu zeichnen. Wie beim Rahmen bestimmen die angegebenen Koordinaten die beiden diagonal gegenüberliegenden Punkte des Feldes.



```
100 REM ****************
110 REM **            **
120 REM **  FLAECHEN  **
130 REM **            **
140 REM ****************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FILL 2,20,30 TO 100,150
180 FILL 2,25,40 TO 180,120
190 FILL 2,55,10 TO 110, 50
200 WAIT 198,255
210 REM
220 GCLEAR
230 FOR X=1 TO 100
240   FILL 2,320*RND(1),200*RND(1) TO 320*RND(1),200*RND(1)
250 NEXT X
260 POKE 198,0 : WAIT 198,255
```

Weitere Anwendungsmöglichkeiten dieser neuen Befehle finden Sie im folgenden Paragraphen, in dem unser CAD-Zeichner erweitert wird.

2.6.4 CAD-Zeichner: 2. Teil

Im folgenden werden wir unseren CAD-Zeichner aus dem Kapitel 2.5.2 um einige Features erweitern, die uns mit den neuen Befehlen nun möglich sind. Wir werden hier nicht das ganze Programm auflisten, sondern nur diejenigen Zeilen, die neu hinzukommen, um nicht unnötig Platz zu verschwenden. Im Anhang wird Ihnen dann ein vollständiges Listing mit kompletter Kurzanleitung geboten.

Vergessen Sie also nicht: Die folgenden BASIC-Zeilen sind nur ein Teilstück des Gesamtprogrammes und nicht für sich allein lauffähig!

```
100 REM **************************
110 REM **                      **
120 REM **  CAD-ZEICHNER TEIL 2 **
130 REM **                      **
140 REM **************************
150 REM
470 IF A$="L" THEN FZ=1 : GOSUB 1450 :REM LINIE
480 IF A$="R" THEN FZ=2 : GOSUB 1450 :REM RECHTECK
490 IF A$="B" THEN FZ=3 : GOSUB 1450 :REM BOX
500 IF A$="N" THEN GOSUB 5040 :REM NETZ ZEICHNEN
1430 REM
1440 REM OBJEKTE ZEICHNEN:
```

```
1450 X1=X : Y1=Y :REM STARTKOORDINATEN
1460 X2=X : Y2=Y :REM ENDKOORDINATEN
1470 ZM=2 : FL=0 :REM FLAG ZURUECKSETZEN
1480 GOSUB 1730  :REM LINIE, FRAME, FILL
1490 FL=ABS(FL-1)
1500 FOR Z=1 TO 30
1510   GET A$ : IF A$="" THEN NEXT Z : GOTO 1480
1520 IF FL=1 THEN GOSUB 1730  :REM WIEDER ZURUEKSETZEN
1530 A=ASC(A$)
1540 IF A=17  THEN GOSUB 1300 :REM RUNTER
1550 IF A=145 THEN GOSUB 1240 :REM HOCH
1560 IF A=29  THEN GOSUB 1340 :REM RECHTS
1570 IF A=157 THEN GOSUB 1390 :REM LINKS
1580 IF A=133 THEN ZM=0 : GOTO 1740  :REM F1
1590 IF A=137 THEN ZM=1 : GOTO 1740  :REM F2
1600 IF A=134 THEN ZM=0 : GOSUB 1740 :REM F1
1610 IF A=138 THEN ZM=1 : GOSUB 1740 :REM F2
1620 IF A= 20 THEN X=X1 : Y=Y1 : RETURN :REM DEL => NICHT ZEICHNEN
1630 IF A= 19 THEN CF=ABS(CF-1)
1690 REM
1700 X2=X : Y2=Y
1710 GOTO 1470
1720 REM
1730 ON FZ GOTO 1800,1900,1900 :REM SPRUNG ZU: LINIE, FRAME, FILL
1740 ON FZ GOTO 1800,1900,2000 :REM SPRUNG ZU: LINIE, FRAME, FILL
1770 REM
1780 REM
1790 REM LINIE ZEICHNEN/LOESCHEN:
1800 PLOT ZM,X1,Y1 TO X2,Y2
1810 RETURN
1870 REM
1880 REM
1890 REM RAHMEN ZEICHNEN/LOESCHEN:
1900 FRAME ZM,1,X1,Y1 TO X2,Y2
1910 RETURN
1970 REM
1980 REM
1990 REM BOX ZEICHNEN/LOESCHEN:
2000 FILL ZM,X1,Y1 TO X2,Y2
2010 RETURN
5000 REM
5010 REM
5020 REM NETZ BZW. RASTER ZEICHNEN
5030 REM
5040 GMODE ,8,161,250 :REM TEXTFENSTER OEFFNEN
5050 PRINT CHR$(147)  :REM TEXT LOESCHEN
5060 POS= 0,20
5070 PRINT "PUNKT-/LINIEN-NETZ (P/L)"
5080 PRINT "NETZPUNKTE-EINHEIT"
5090 POS= 26,20
5100 INPUT PF$ : IF PF$<>"L" AND PF$<>"P" THEN 5090
```

```
5110 POS= 26,21
5120 INPUT NE$ : IF VAL(NE$)<2 THEN 5110
5130 GMODE ,1  :REM GRAPHIK WIEDER VOLL
5140 NE=VAL(NE$)
5150 IF PF$="L" THEN 5220
5160 REM
5170 REM PUNKTNETZ:
5180 FOR XN=0 TO 319 STEP NE
5190   FOR YN=0 TO 199 STEP NE
5200     PLOT 2,XN,YN : PLOT 2,XN+1,YN
5210   NEXT YN
5220 NEXT XN
5230 RETURN
5240 REM
5250 REM PUNKTNETZ:
5260 FOR XN=0 TO 319 STEP NE
5270   PLOT 2,XN,0 TO XN,199
5280 NEXT XN
5290 FOR YN=0 TO 199 STEP NE
5300   PLOT 2,0,YN TO 319,YN
5310 NEXT YN
5320 RETURN
```

Hier eine kurze Bedienungsanleitung: Durch folgende Funktionen wurde der CAD-Zeichner in dem oben aufgelisteten Programmteil erweitert:

- L - Linie zeichnen
- R - Rechteck zeichnen
- B - Ausgefülltes Rechteck (Box) zeichnen
- N - Rasternetz in die Graphik einzeichnen

Wollen Sie eine Linie zeichnen, so bringen Sie den kleinen Graphikcursor an die Stelle auf dem Bildschirm, an der diese Linie beginnen soll. Betätigen Sie nun die L-Taste. Jetzt befinden Sie sich im sogenannten Linien-Modus. Sie haben die Möglichkeit, den Endpunkt der zu zeichnenden Linie nun mittels Cursortasten festzulegen. Dabei zeigt Ihnen stets eine blinkende Linie ihre momentane Lage, die sie einnnähme, wenn Sie den Befehl zum Zeichnen gäben. Im Linien-Modus stehen Ihnen die folgenden Befehle zur Verfügung:

CRSR hoch	-	Linienendpunkt hoch
CRSR hinunter	-	Linienendpunkt hinunter
CRSR rechts	-	Linienendpunkt rechts
CRSR links	-	Linienendpunkt links
clr/home	-	Cursorgeschwindigkeit
DEL	-	Linienmodus abbrechen
f1	-	Linie zeichnen
f2	-	Linie löschen
f3	-	Linie zeichnen und mit gleichem Startpunkt die nächste Linie
f4	-	Linie löschen und mit gleichem Startpunkt die nächste Linie

Wie im normalen Zeichenmodus können Sie auch hier die Geschwindigkeit des Cursors mit der Taste clr/home zwischen 1 und 10 Punkten Fortbewegung pro Tastendruck wechseln.

Falls Sie sich bei dem Einstieg in den Linienmodus geirrt haben oder doch keine Linie mehr zeichnen wollen, so betätigen Sie die DEL-Taste und schon wird alles rückgängig gemacht.

Sie haben nun den Endpunkt ihrer Linie mithilfe der Cursortasten gefunden und wollen tatsächlich eine Linie auf dem Bildschirm abbilden. In diesem Fall bieten sich Ihnen die oben aufgeführten insgesamt 4 Möglichkeiten.

Zeichnen oder löschen Sie die positionierte Linie mit f1 oder f2, so gelangen Sie wieder außerhalb des Linienmodus. Der Graphikcursor befindet sich dann exakt am Linienendpunkt. Wollen Sie direkt im Anschluß an diese gerade gezeichnete Linie eine weitere zeichnen, so bewegen Sie den Cursor nicht, sondern drücken gleich wieder "L".

Sie möchten zwar die Linie zeichnen bzw. löschen, jedoch gleich wieder vom selben Startpunkt der alten Linie eine neue bearbeiten, also quasi Radien erzeugen, so verwenden Sie die Tasten f3 bzw. f4. Sie bleiben im Linien-Modus und können hier gleich Ihre neue Linie positionieren.

Die Taste "R" - im Hauptmodus betätigt - bringt Sie in den Rechteck-Modus. Im Prinzip haben Sie hier dieselben Möglichkeiten, die Ihnen im Linien-Modus zur Verfügung stehen (Sie werden unten sehen, daß es sich hier um den gleichen Programmteil handelt). Wenn Sie in die obigen Ausführungen die Wörter "Linie" durch "Rechteck", "Linienendpunkt" durch "Ecke des Rechtecks" ersetzten, so sind Sie auf dem

richtigen Wege. In der Tat ist es so, daß Sie durch den Einstieg in den Rechteck-Modus einen Eckpunkt dieses Objektes durch die momentane Cursorposition festlegen. Bewegen Sie nach dem "R" nun den Cursor in zwei Richtungen, so erscheint auf dem Bildschirm ein blinkendes Rechteck, dessen zweiten Eckpunkt (diagonal gegenüber dem ersten) Sie jetzt mit den Cursortasten festlegen können. Probieren Sie es doch einfach einmal aus. Bilder sagen mehr als tausend Worte.

Wollen Sie sogar ein ausgefülltes Rechteck zeichnen (Box), so wählen Sie statt "R" einfach "B". Hierbei können Sie alles aus dem vorherigen Abschnitt übernehmen.

Es bleibt ein Kommando: Das Kommando zur Erzeugung eines Rasternetzes über den gesamten Bildschirm ("N"). Haben Sie diese Taste im Hauptmodus angewählt, so erscheint am unteren Teil des Bildschirms ein kleines Textfenster, in dem Sie nach der Art des Netzes und seiner Einteilung gefragt werden. Hier können Sie - je nachdem ob Sie L oder P eingeben - bestimmen, ob sich Ihr Netz aus einfachen Punkten oder aus Linien zusammensetzen soll. Haben Sie Ihre Eingabe mit <return> bestätigt, dann geben Sie noch schnell die Anzahl der Punkte zwischen zwei Netzlinien oder -punkten an und schon sehen Sie zu, wie Ihr Rechner den Bildschirm "vernetzt".

Dieses Kommando sollte nach Möglichkeit nur einmal und nur am Anfang einer Arbeit gegeben werden, da ansonsten alles bisher gezeichnete durch die Linien oder Punkte überzeichnet wird. Wollte man dies verhindern, dann müßte man von jedem gezeichneten Objekt (außer dem Netz selbst natürlich) eine Kopie in der zweiten Graphikseite anfertigen. D.h. jedesmal, wenn ein Objekt gezeichnet wird, sei es Punkt, Linie, Rechteck oder ausgefülltes Rechteck, müßten Sie (verdeckt natürlich) auf die zweite Graphikseite umschalten und es an der gleichen Stelle noch einmal zeichnen. Sie brauchen natürlich den Graphikcursor und die blinkenden Linien oder Rechtecke nicht in die zweite Seite kopieren. Diese Änderungen sind recht einfach in nur wenigen zusätzlichen Zeilen zu realisieren. Wenn Sie dann einmal das Netz wieder fortnehmen möchten oder z.B. die Netzeinheit ändern, dann brauchen Sie nur die gesamte zweite Seite in die erste zu kopieren. Wie dies mit dem GCOMB-Befehl funktioniert, werden Sie in späteren Kapiteln erfahren. Vielleicht versuchen Sie sich ja einmal an einer solchen Programmänderung.

Damit Sie auch das Programm selbst besser verstehen, hier eine Beschreibung der wichtigsten Teile:

In den Zeilen 470-500 wurden die neuen Tasten implementiert. Wie Sie sehen, wird bei den Modi Linie, Rechteck und Box zur selben Routine (nach Zeile 1450) gesprungen. Allein die Variable FZ (Zeichenflag) bestimmt die Art des zu zeichnenden Objektes. Diese ab Zeile 1450 beginnende Routine hat einige Ähnlichkeit mit der Hauptroutine zum Cursorblinken und der Tastaturabfrage. Statt eines einzelnen Punktes blinkt dann aber durch Zeile 1480 je nach FZ eine Linie oder ein Rechteck. In dieser Zeile wird nämlich ein Unterprogrammsprung auf eine Zeile unternommen (Zeile 1730), in der ein ON FZ GOTO auf die Befehle "Linie zeichnen" oder "Rechteck zeichnen" springt. Vorher wurde der Zeichenmodus auf 2 (invertieren) gesetzt, um keine bereits vorhandenen Zeichnungen zu zerstören (s. Hauptroutine).

Wurde nun eine Taste betätigt, wird einzeln jede erlaubte Taste abgefragt und zu den verschiedenen dafür zuständigen Routinen gesprungen. Die 4 Cursorroutinen stammen aus dem ersten Programmteil. In den Zeilen 1580-1610 wird dann - wie gerade beschrieben - wieder auf ein ON FZ GOTO gesprungen, um die jeweils richtige Figur zu zeichnen (ZM=0) oder zu löschen (ZM=1).

Ab Zeile 5040 beginnt die Netz-Routine. Hierzu wird zunächst in genau dieser Zeile ein kleines Textfenster geöffnet, um dann die beiden erforderlichen Parameter abzufragen. Das Zeichnen des Netzes übernehmen dann FOR...NEXT-Schleifen.

Mit diesem Programm können Sie nun schon recht prächtige Graphiken entwerfen. Vielleicht versuchen Sie ja einmal, die Cursorsteuerung statt von der Tastatur, vom Joystick aus vorzunehmen, denn die bisherige Methode ist doch auf die Dauer sehr umständlich. Der später besprochene Befehl IF#...THEN... wird Ihnen dabei sicher eine Hilfe sein.

2.7 Kreise und alles, was (fast) rund ist

Wir können schon sehr viel. Wir könnten auch Kreise zeichnen. Doch was ist eine umständliche und zeitraubende Kreisroutine in BASIC (wir werden später noch die verschiedenen Möglichkeiten der Kreiszeichnung durchsprechen) gegenüber einem schnellen und von BASIC aus einfach zu bedienenden Kreisprogramm? Das hat sich der Autor der Supergraphik auch gesagt und einen Kreisbefehl geschaffen.

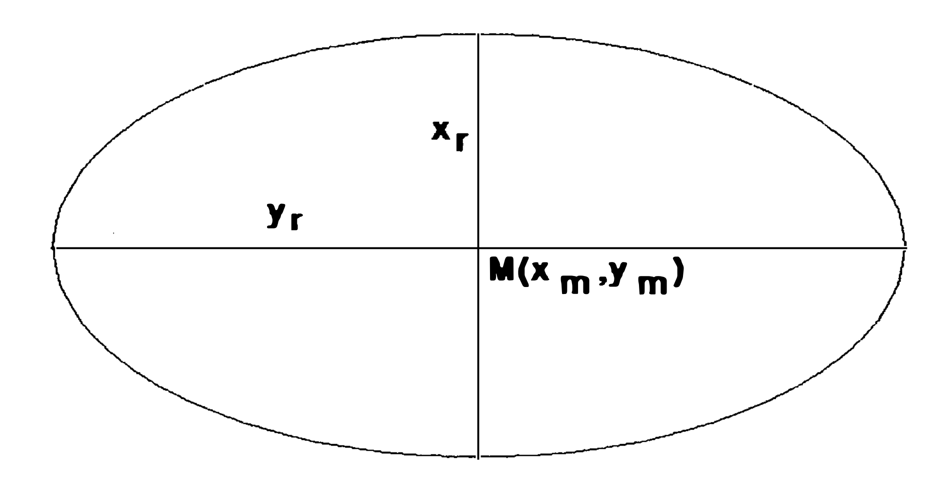
2.7.1 Der CIRCLE-Befehl

CIRCLE

Befehl:	CIRCLE zm,,xm,ym,xr,yr
Beispiel:	CIRCLE 0,,160,100,80,90
Parameter:	zm: Zeichenmodus
xm:	x-Mittelpunktskoordinate
ym:	y-Mittelpunktskoordinate
xr:	x-Radius
yr:	y-Radius
Funktion:	Setzen (Loeschen etc.) einer Ellipse, Kreis

Mit diesem Befehl werden Ihnen Welten eröffnet:
Kreise und Ellipsen (und wie Sie später sehen werden noch eine Reihe weiterer Figuren) können durch ein "Fingerschnippen" Ihrerseits auf den Bildschirm gezaubert werden.
xm,ym geben dabei die Koordinaten des Mittelpunktes der Figur an, xr den Radius in x-Richtung und yr den in y-Richtung. Sind die letzten beiden Werte gleich, so entsteht ein Kreis (außer in Multicolor), ansonsten eine Ellipse. Unter Umständen können Verzerrungen durch die mangelhafte Einstellung Ihres Bildschirms oder durch die speziellen Eigenheiten Ihres Druckers entstehen.

Wie das alles mit x-Radius, y-Radius, Mittelpunkt pi, pa, po zusammenhängt, soll Ihnen das folgende kleine Programm demonstrieren:



```
100 REM **************
110 REM **          **
120 REM **  CIRCLE  **
130 REM **          **
140 REM **************
150 REM
160 GMODE ,8,193,250 : GCLEAR : SCOL= 0,0
170 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
180 POS= 10,23
190 PRINT "  XM,  YM,  XR,  YR";
200 POS= 0,24
210 PRINT "CIRCLE 0,,    ,    ,    ,";
220 XM=160 : YM=100
230 XR= 20 : YR= 50
240 S=1              :REM SCHRITTWEITE
250 REM
260 REM HAUPTSCHLEIFE:
270 REM
280 GMODE ,1 : CIRCLE ,,XM,YM,XR,YR
290 GMODE ,8,193,250
300 PLOT ,XM,YM : PLOT ,XM+1,YM :REM MITTELPUNKT ZEICHNEN
310 REM
320 POS= 10,24 : PRINT XM;
330 POS= 15,24 : PRINT YM;
340 POS= 20,24 : PRINT XR;
350 POS= 25,24 : PRINT YR;
360 REM
370 GET A$ : IF A$="" THEN 370
380 A=ASC(A$)
390 IF A= 17  THEN 530          :REM CRSR RUNTER
400 IF A=145  THEN 580          :REM CRSR HOCH
410 IF A= 29  THEN 630          :REM CRSR RECHTS
420 IF A=157  THEN 680          :REM CRSR LINKS
430 IF A$="W" THEN FL=ABS(FL-1) :REM FLAG WECHSELN
440 IF A$="S" THEN S=S*10/S^2   :REM ZWISCHEN 10 UND 1 WECHSELN
450 IF A$="Q" THEN END          :REM ENDE
460 GOTO 370
470 REM
480 REM
490 REM AUSFUEHRUNGEN:
500 REM
510 REM RUNTER:
520 REM
530 XD= 0 : YD= 1 :REM KOORDINATEN VERSCHIEBUNGEN
540 GOTO 740
550 REM
560 REM HOCH:
570 REM
580 XD= 0 : YD=-1 :REM KOORDINATEN VERSCHIEBUNGEN
590 GOTO 740
```

```
600 REM
610 REM RECHTS:
620 REM
630 XD= 1 : YD= 0 :REM KOORDINATEN VERSCHIEBUNGEN
640 GOTO 740
650 REM
660 REM LINKS:
670 REM
680 XD=-1 : YD= 0 :REM KOORDINATEN VERSCHIEBUNGEN
690 GOTO 740
700 REM
710 REM
720 REM NEUEN KREIS ZEICHNEN:
730 REM
740 GMODE ,1
750 CIRCLE 1,,XM,YM,XR,YR :REM ALTEN KREIS LOESCHEN
760 GMODE ,8,193,250
770 PLOT 1,XM,YM : PLOT 1,XM+1,YM :REM ALTEN MITTELPUNKT LOESCHEN
780 IF FL=0 THEN 850      :REM RADIEN AENDERN
790 REM
800 REM MITTELPUNKT AENDERN:
810 XM=ABS(XM+XD*S) : YM=ABS(YM+YD*S)
820 GOTO 280
830 REM
840 REM RADIEN AENDERN:
850 XR=ABS(XR+XD*S) : YR=ABS(YR-YD*S)
860 GOTO 280
```

Für eine Befehlsvorstellung scheint dieses Programm vielleicht ein bißchen sehr groß geraten. Doch es zeigt Ihnen direkt, wie die einzelnen Parameter des CIRCLE-Befehls wirken. Während sich nämlich auf dem Graphikbildschirm die aktuelle Ellipse mit den aktuellen Eigenschaften befindet, wird in den unteren beiden Zeilen des Bildschirms direkt der Befehl ausgegeben, der notwendig wäre, um diese Ellipse auf den Bildschirm zu zeichnen. Dies wird durch Einschalten eines kleinen Textfensters (Zeile 160) erreicht. Sie können nun durch folgende Tasten die auf dem Bildschirm befindliche Ellipse verändern:

CRCR runter	- Mittelpunkt nach unten verschieben bzw. y-Radius verringern
CRSR hoch	- Mittelpunkt nach oben verschieben bzw. y-Radius erhöhen
CRSR rechts	- Mittelpunkt nach rechts verschieben bzw. x-Radius erhöhen
CRSR links	- Mittelpunkt nach links verschieben bzw. x-Radius verringern
W	- Wechsel zwischen: - Mittelpunktverschiebung - Radiusänderung
S	- Schrittweite der Änderungen in Punkten
Q	- Programmabbruch

Wenn Sie das obige Programm starten, dann können Sie durch Betätigung der Cursortasten in der beschriebenen Art und Weise die beiden Radien der Ellipse verändern. Bei jedem Druck auf die Tasten vergrößern oder verkleinern sie sich um jeweils einen Punkt bzw. um 10 Punkte, wenn vorher durch die "S"-Taste die Schrittweite erhöht wurde. Bei jeder Änderung wird der in der untersten Zeile abgebildete Befehl aktualisiert mit den neuesten Werten.

Wollen Sie nicht die Größe der Radien verändern, so betätigen Sie die Taste "W". Nun wird gewechselt: Bei jeder Betätigung der Cursortasten wird nun der Mittelpunkt der Ellipse (des Kreises) bewegt. Ein weiterer Druck auf "W" leitet wieder den Wechsel zurück ein. Wollen Sie das Programm beenden, so genügt ein Druck auf "Q".

Sicher haben Sie bereits das Flackern des Bildschirms bemerkt, wenn Sie einen Parameter verändern. Dies rührt daher, daß in den Zeilen 280/290 und 740-760 vor dem Zeichnen bzw. Löschen einer Ellipse das Textfenster aus- und später wieder eingeschaltet wird. Dies ist notwendig, um ein noch größeres Flackern zu verhindern. Während des CIRCLE-Befehls wird nämlich der Interrupt (s.o.) ausgeschaltet, womit auch die Sache mit dem Textfenster zusammenbricht.

Das Programm ist allgemein möglichst einfach und übersichtlich geschrieben. Es sollte also kein großes Problem darstellen, es zu verstehen. Vielleicht sollten doch noch ein paar Kleinigkeiten erläutert werden.

Da ist zum Beispiel die Angelegenheit mit dem Wechseln der Schrittweite in Zeile 440. Die Variable S wechselt durch die Formel S=S*10/S^2 ständig zwischen 1 und 10. Rechnen Sie doch einfach einmal nach. Ähnliches bewirkt die inzwischen schon bekannte Formel in Zeile 430. Hier wechselt FL immer zwischen 0 und 1.

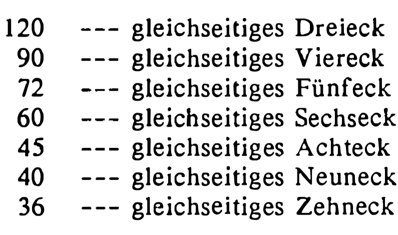
In den Zeilen 510-690 werden die Cursorbewegungen vollzogen, bzw. hier werden die Werte festgelegt, die (multipliziert mit S) zu den Mittelpunktskoordinaten bzw. den Radien addiert werden (Zeilen 810 und 850). Um Ihnen die Lage des Mittelpunktes zu verdeutlichen, wird er noch einmal getrennt eingezeichnet.

Polygone:

Befehl:	CIRCLE zm,sl,xm,ym,xr,yr	
Beispiel:	CIRCLE 0,120,160,100,80,90	
Parameter:	zm:	Zeichenmodus
	sl:	Seitenlänge (0-255)
	xm:	x-Mittelpunktskoordinate
	ym:	y-Mittelpunktskoordinate
	xr:	x-Radius
	yr:	y-Radius
Funktion:	Setzen (loeschen etc.) eines beliebigen Vielecks	

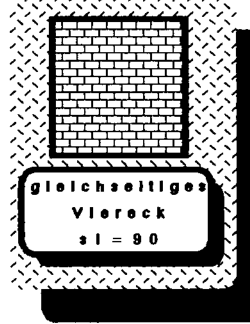
Mit der sl-Ergänzung des CIRCLE-Befehls können Sie das Zeichnen von Kreisen und Ellipsen ausweiten auf Quadrate, Rechtecke, 5-, 6-,... Ecke. Dabei gibt sl die jeweilige Kantenlänge in Grad des überspannenden Bogens an. Was ist sich darunter vorzustellen?

Genau genommen können Sie mit sl angeben, in welchem Gradabstand die einzelnen Punkte einer jeweiligen Ellipse gezeichnet werden sollen. Normalerweise wird alle 2 Grad ein Punkt berechnet und die entstehenden Punkte mit einer Linie verbunden (Vielleicht erinnern Sie sich noch an die Einführung zum Linien-Kapitel. Dort wurde erläutert, weshalb alle Figuren aus Linien angenähert werden. Dies ist auch hier beim Kreis so). Vergrößert man nun diesen Abstand, so nimmt die Genauigkeit der Zeichnung stetig ab, bis direkt ein Vieleck zu beobachten ist. Interessant sind besonders dabei die folgenden Werte:

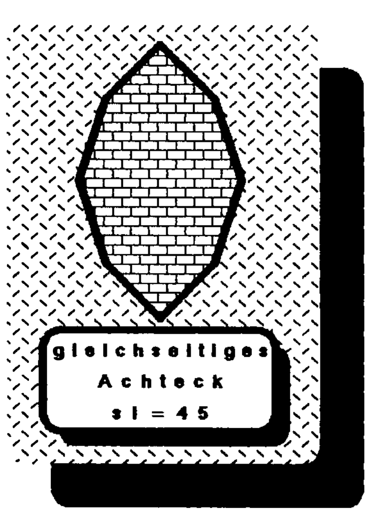


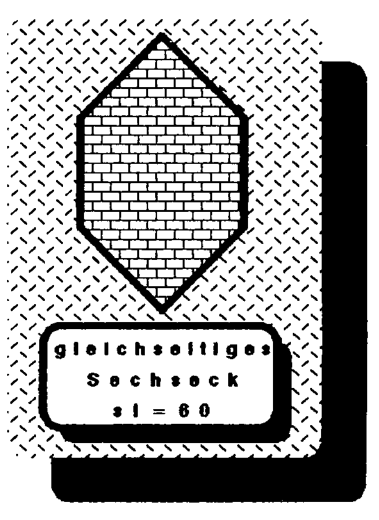
120 --- gleichseitiges Dreieck
90 --- gleichseitiges Viereck
72 --- gleichseitiges Fünfeck
60 --- gleichseitiges Sechseck
45 --- gleichseitiges Achteck
40 --- gleichseitiges Neuneck
36 --- gleichseitiges Zehneck







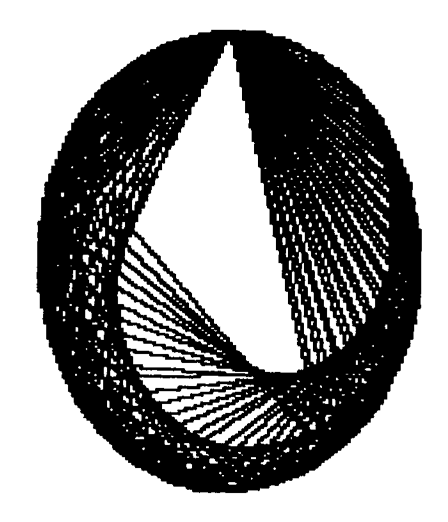




Dabei sind jeweils größere Werte immer noch Gebilde mit der entsprechenden Eckenzahl. Ein CIRCLE-Befehl mit dem sl-Wert 100 z.B. produziert immer noch ein Viereck. Die oben angegebenen Werte stellen die Übergänge von n+1- zum n-Eck dar.

Das folgende Programm zeigt Ihnen, wie sich sl auf den CIRCLE-Befehl auswirkt:

```
100 REM ****************
110 REM **            **
120 REM **  POLYGONE  **
130 REM **            **
140 REM ****************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR SL=2 TO 100 STEP 2
180   CIRCLE 2,SL,160,100,60,90
190   CIRCLE 2,SL,160,100,60,90
200 NEXT SL
210 CIRCLE 2,SL,160,100,60,90
220 WAIT 198,255
```



Bevor wir uns weiter mit diesen Möglichkeiten des CIRCLE-Befehls beschäftigen, wollen wir nun auch das letzte aus ihm herausholen:

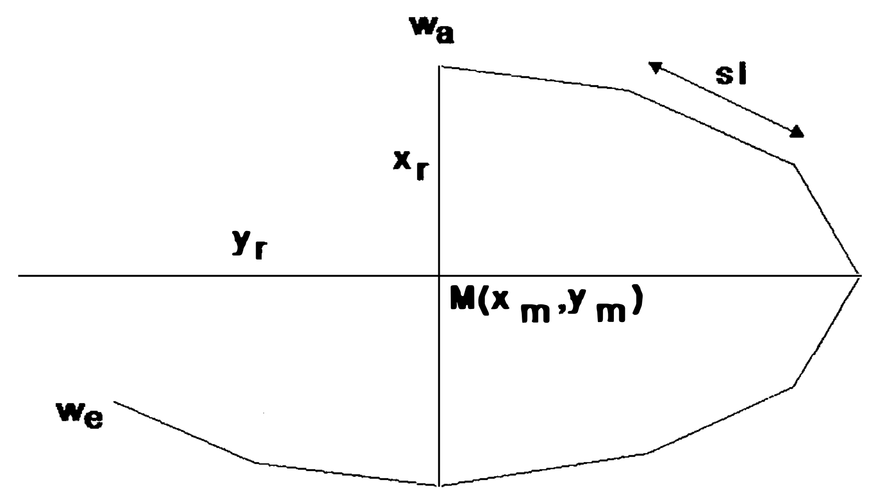
Bögen:

Befehl:	CIRCLE zm,(sl),xm,ym,xr,yr,wa,we
Beispiel:	CIRCLE 0,2,160,100,80,90,30,130
Parameter:	zm: Zeichenmodus
	sl: Seitenlänge (0-255)
	xm: x-Mittelpunktskoordinate
	ym: y-Mittelpunktskoordinate
	xr: x-Radius
	yr: y-Radius
	wa: Anfangswinkel (0-360)
	we: Endwinkel (0-360)
Funktion:	Setzen (Loeschen etc.) von Kreis-, Ellipsen- oder Vieleck-Bögen

Die Spitze des CIRCLE-Befehls wird durch die Angabe des Start- und Endwinkels der Ellipse bzw. des Vielecks erreicht. Sie geben also an, von welcher Gradzahl an die Figur gezeichnet wird und wo Sie enden soll. Auf diese Weise erhalten Sie z.B. Halbkreise, Viertelellipsen etc. Sie können auch nur einen einzigen Punkt berechnen lassen und ihn als Ausgangspunkt für weitere Zeichnungen verwenden (z.B. zum Mittelpunkt mittels PLOT TO ..., um einen Radius zu zeichnen etc.).

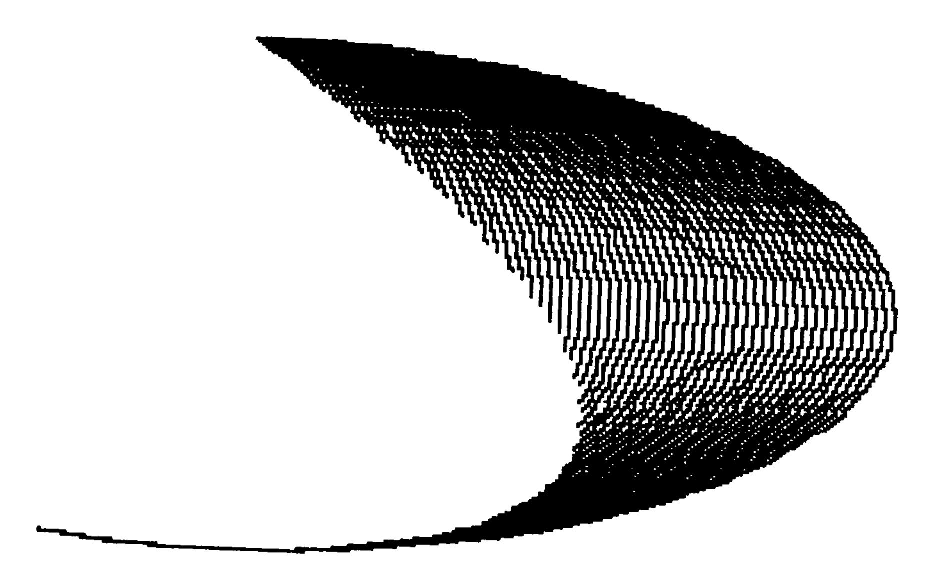
```
10 GMODE 0,1 : GCLEAR
20 CIRCLE ,,159,99,60,80,0,180
30 WAIT 198,255
```

Hier wird Ihnen die Benutzung des Befehls anhand einer Halbellipse mit dem Mittelpunkt auf der Mitte des Bildschirms demonstriert. Dies ist einer der "Spiel"-Befehle, deren Möglichkeiten erst im Laufe des Probierens voll klar werden. Was halten Sie beispielsweise von diesem Programm:



```
10 GMODE 0,1 : GCLEAR
20 FOR X=1 TO 200 STEP 3
30   CIRCLE ,,90,99,X,99,0,X
40 NEXT X
```

Sollte Ihr Gebilde einmal größer als der Bildschirm werden, so besteht kein Grund zur Beunruhigung. Der Rechner hört früh genug auf.



Das folgende kleine Programm ist eine Erweiterung des weiter oben vorgestellten und nutzt nun alle Möglichkeiten des CIRCLE-Befehls aus. Hier können Sie nach Herzenslust probieren und die Wirkung der einzelnen Parameter studieren:

```
100 REM *************************
110 REM **                     **
120 REM **  CIRCLE - KOMPLETT  **
130 REM **                     **
140 REM *************************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
180 POS= 9,23
190 PRINT " SL, XM, YM, XR, YR, WA, WE";
200 POS= 0,24
210 PRINT "CIRCLE 0,  ,  ,  ,  ,  ,  ,";
```

```
220 KO=1E-7          :REM KORREKTURSUMMAND
230 SL=  2           :REM SEITENLAENGE
240 XM=160 : YM=100 :REM MITTELPUNKT
250 XR= 20 : YR= 50 :REM RADIEN
260 WA=  0 : WE=360 :REM START-/ENDWINKEL
270 S=1              :REM SCHRITTWEITE
280 FL=1             :REM FLAG
290 REM
300 REM HAUPTSCHLEIFE:
310 REM
320 CIRCLE ,SL,XM,YM,XR,YR,WA,WE
330 GMODE ,8,193,250
340 PLOT ,XM,YM : PLOT ,XM+1,YM :REM MITTELPUNKT ZEICHNEN
350 REM
360 REM WERTE AUSGEBEN:
370 PO=13
380 VA=SL : GOSUB 1090
390 VA=XM : GOSUB 1090
400 VA=YM : GOSUB 1090
410 VA=XR : GOSUB 1090
420 VA=YR : GOSUB 1090
430 VA=WA : GOSUB 1090
440 VA=WE : GOSUB 1090
450 REM
460 GET A$ : IF A$="" THEN 460
470 A=ASC(A$)
480 IF A= 17  THEN 620          :REM CRSR RUNTER
490 IF A=145  THEN 670          :REM CRSR HOCH
500 IF A= 29  THEN 720          :REM CRSR RECHTS
510 IF A=157  THEN 770          :REM CRSR LINKS
520 IF A$="W" THEN FL=FL+1 : IF FL=5 THENFL=1 :REM FLAG WECHSELN
530 IF A$="S" THEN S=S*10/S^2   :REM ZWISCHEN 10 UND 1 WECHSELN
540 IF A$="Q" THEN END          :REM ENDE
550 GOTO 460
560 REM
570 REM
580 REM AUSFUEHRUNGEN:
590 REM
600 REM RUNTER:
610 REM
620 XD= 0 : YD= 1 :REM KOORDINATEN VERSCHIEBUNGEN
630 GOTO 830
640 REM
650 REM HOCH:
660 REM
670 XD= 0 : YD=-1 :REM KOORDINATEN VERSCHIEBUNGEN
680 GOTO 830
690 REM
700 REM RECHTS:
710 REM
720 XD= 1 : YD= 0 :REM KOORDINATEN VERSCHIEBUNGEN
```

```
730 GOTO 830
740 REM
750 REM LINKS:
760 REM
770 XD=-1 : YD= 0 :REM KOORDINATEN VERSCHIEBUNGEN
780 GOTO 830
790 REM
800 REM
810 REM NEUEN KREIS ZEICHNEN:
820 REM
830 GMODE ,1
840 CIRCLE 1,SL,XM,YM,XR,YR,WA,WE :REM ALTEN KREIS LOESCHEN
850 PLOT 1,XM,YM : PLOT 1,XM+1,YM :REM ALTEN MITTELPUNKT LOESCHEN
860 ON FL GOTO 930,890,970,1030   :REM RADIEN AENDERN
870 REM
880 REM MITTELPUNKT AENDERN:
890 XM=INT(ABS(XM+XD*S)+KO) : YM=INT(ABS(YM+YD*S)+KO)
900 GOTO 320
910 REM
920 REM RADIEN AENDERN:
930 XR=INT(ABS(XR+XD*S)+KO) : YR=INT(ABS(YR-YD*S)+KO)
940 GOTO 320
950 REM
960 REM START-/ENDWINKEL AENDERN:
970 WA=INT(ABS(WA+XD*S)+KO) : WE=INT(ABS(WE-YD*S)+KO)
980 IF WA>360 THEN WA=0
990 IF WE>360 THEN WE=0
1000 GOTO 320
1010 REM
1020 REM SEITENLAENGE AENDERN:
1030 SL=INT(ABS(SL+XD*S-YD*S)+KO)
1040 GOTO 320
1050 REM
1060 REM
1070 REM WERTEAUSGABE:
1080 REM
1090 VA$=STR$(VA)
1100 POS= PO-4,24 : PRINT "    ";
1110 POS= PO-LEN(VA$),24 :PRINT RIGHT$(VA$,LEN(VA$)-1);
1120 PO=PO+4             :REM NAECHSTE SPALTE
1130 RETURN
```

2.7.2 Die hochaufgelöste BASIC-Uhr

Im folgenden möchten wir Ihnen eine sehr schöne und tatsächlich auch nutzbringende Anwendung vorstellen. Für alle diejenigen unter Ihnen, denen ihr Morgenmuffeltum ständig Ärger und Sorgen bereitet, haben wir eine kleine Spezialität vorbereitet.

Hand aufs Herz! Der allmorgendliche, weckertötende Zorn reißt zunehmend große Löcher in Ihr monatliches Uhrenbudget. Hier bieten wir Ihnen die Lösung:

Ihr Commodore-Computer übernimmt heldenhaft sämtliche Funktionen Ihres Weckers. Wenn Sie also morgens zornentbrannt Ihren Wecker suchen, um ihm den verstummenden Schlag zu verabreichen, lächelt Sie bereits Ihr Rechner an. Sie werden sich schnell eines anderen besinnen und den Schlag zu einem zärtlichen Tasten"klick" dämpfen - entweder aus Liebe zu Ihrem Computer oder eingedenk der drohenden Werkstattrechnung.

Wenn Sie nun darauf brennen, die Lösung all Ihrer Probleme vor Augen zu haben, hier ist sie:

```
100 REM ****************************
110 REM **                        **
120 REM **  DIE UHRZEIT, BITTE!   **
130 REM **                        **
140 REM ****************************
150 REM
160 GMODE 0,5 : GCLEAR : SCOL= 0,0
170 PI=3.14159265
180 REM
190 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
200 PRINT "GEBEN SIE DIE UHRZEIT AN!"
210 PRINT
220 INPUT "STUNDE, MINUTE, SEKUNDE ";ST$,MI$,SE$
230 TI$=ST$+MI$+SE$  :REM UHRZEIT STELLEN
240 INPUT "DATUM: BITTE TAG ANGEBEN";TA$
250 TA=VAL(TA$)
260 PRINT
270 INPUT "WECKZEIT: STUNDE, MINUTE";WS$,WM$
280 REM
290 WZ$=WS$+WM$      :REM WECKZEIT MERKEN
300 REM
310 GMODE 0,1        :REM GRAPHIK EIN
320 REM
330 REM
340 REM UHR ZEICHNEN:
```

```
350 REM
360 TEXT ,"TIMERUNNER",120,10,0
370 REM
380 CIRCLE ,,160,100,80,80
390 CIRCLE ,,160,100,60,60
400 REM
410 REM TEILSTRICHE:
420 REM
430 Y=2
440 FOR W=0 TO 359 STEP 360/(12*4)     :REM KREIS IN 48 TEILE
450   Y=Y+1 : IF Y=4 THEN Y=0          :REM JE 4 STRICHE ABZAEHLEN
460   FOR X=0 TO 4+4*INT(Y/3)          :REM JEDEN 4. STRICH GROSS
470     CIRCLE ,,160,100,62-X,62-X,W,W :REM KREISPUNKT BERECHNEN
480   NEXT X
490 NEXT W
500 REM
510 REM ZIFFERN EINZEICHNEN:
520 REM
530 X=0 : EN=2*PI/12                :REM SCHRITTWEITE
540 FOR W= -2*EN TO 9*EN STEP EN
550   X=X+1 : X$=STR$(X)            :REM ZIFFER
560   TEXT ,RIGHT$(X$,LEN(X$)-1),70*COS(W)+156,70*SIN(W)+104,0
570 NEXT W
580 REM
590 REM TAGESANZEIGE:
600 REM
610 FRAME ,2,186,92 TO 210,108
620 REM
630 REM
640 REM
650 REM ZEIGER EINZEICHNEN:
660 REM
670 S1=0 : M1=0 : H1=0 : FL=0
680 REM
690 IF LEFT$(TI$,4)=WZ$ THEN GOSUB 1220 :REM WECKEN
700 REM
710 REM SEKUNDENZEIGER:
720 REM
730 S2=VAL(RIGHT$(TI$,2))    :REM SEKUNDEN HOLEN
740 IF S1=S2 THEN 730        :REM NEUER WERT?
750 ZA=S1 : ZN=S2 : R=78     :REM JA->
760 GOSUB 1120               :REM SEKUNDENZEIGER VERSTELLEN
770 S1=S2                    :REM NEUEN WERT MERKEN
780 REM
790 REM MINUTENZEIGER:
800 REM
810 M2=VAL(MID$(TI$,3,2))    :REM MINUTEN HOLEN
820 IF M1=M2 THEN 730        :REM NEUER WERT?
830 ZA=M1 : ZN=M2 : R=72
840 GOSUB 1120               :REM JA->ZEIGER VERAENDERN
850 M1=M2
```

```
860 REM
870 REM STUNDENZEIGER:
880 REM
890 H2=VAL(LEFT$(TI$,2))/12  :REM STUNDE HOLEN+DURCH 12 TEILEN
900 H2=60*(H2-INT(H2))+M2/12 :REM 24->12 STUNDEN + MINUTENANTEIL
910 IF H1=H2 THEN 730        :REM WERT GEAENDERT
920 ZA=H1 : ZN=H2 : R=40
930 GOSUB 1120               :REM NEUEN ZEIGER SETZEN
940 H1=H2
950 REM
960 REM TAGESANZEIGE:
970 REM
980 IF FL=0 THEN FL=1 : GOTO 1020
990 IF H1<>0 THEN 690
1000 TF=TA : TA=TA+1 : IF TA=32 THEN TA=1
1010 TEXT 2,RIGHT$(STR$(TF),2),190,104,0 :REM ALTEN WERT LOESCHEN
1020 TEXT 2,RIGHT$(STR$(TA),2),190,104,0 :REM NEUEN WERT
1030 GOTO 690
1040 REM
1050 REM
1060 REM ZEIGER NEU ZEICHNEN:
1070 REM UEBERGABE:
1080 REM ZA=ALTE ZEIGERPOSITION
1090 REM ZN=NEUE ZEIGERPOSITION
1100 REM R =ZEIGERLAENGE
1110 REM
1120 IF FL=0 THEN ZW=360/60 : GOTO 1150 :REM ERSTAUFRUF
1130 CIRCLE 4,,160,100,R,R,ZW*ZA,ZW*ZA
1140 PLOT 2, TO 160,100                 :REM ALTEN ZEIGER LOESCHEN
1150 CIRCLE 4,,160,100,R,R,ZW*ZN,ZW*ZN
1160 PLOT 2, TO 160,100                 :REM NEUEN ZEIGER SETZEN
1170 RETURN
1180 REM
1190 REM
1200 REM WECKEN:
1210 REM
1220 VOLUME= 15                :REM VOLLE LAUTSTAERKE
1230 SOUND 1,1,0,1,15,0,0,500 :REM SOUND STIMME 1
1240 SOUND 2,4,0,1,15,0,0,500 :REM SOUND STIMME 2
1250 SOUND 3,4,0,1,15,0,0,500 :REM SOUND STIMME 3
1260 FILTER 0,0,15             :REM FILTEREINSTELLUNG
1270 POKE 198,0                :REM TASTEN LOESCHEN
1280 TUNE 1,12,0,5             :REM TON AUF KANAL 1
1290 TUNE 2,12,1,5             :REM TON AUF KANAL 2
1300 TUNE 3,12,2,5             :REM TON AUF KANAL 3
1310 COLOR= X,X                :REM BILDSCHIRMFARBEN WECHSELN
1320 X=X+1 : IF X=16 THEN X=0
1330 GET A$ : IF A$="" THEN 1280 :REM AUF TASTE PRUEFEN
1340 REM
1350 REM TON AUS:
1360 VOLUME= 0
```

```
1370 SOUND 1,0,0,0,0,0,1
1380 SOUND 2,0,0,0,0,0,1
1390 SOUND 3,0,0,0,0,0,1
1400 FILTER 7
1410 COLOR= 0,0                    :REM FARBEN WIEDER NORMAL
1420 RETURN
```

TIMERUNNER

Wenn Sie dieses Programm geladen und gestartet haben, werden Sie zunächst einmal nach der aktuellen Uhrzeit gefragt - Ihre Uhr muß schließlich erst gestellt werden. Die Uhrzeit geben Sie bitte in folgendem Format ein:

hh,mm,ss z.B.: 23,04,59

Geben Sie also Stunde (hh), Minute (mm) und Sekunde (ss) stets zweistellig und jeweils mit Komma abgetrennt ein. Nur so kann Ihre Uhr richtig gehen. Sobald Sie nun <return> drücken, wird die genaue Uhrzeit übernommen.

Als Nächstes sollten Sie das Datum des heutigen Tages eingeben (hierbei ist es gleichgültig, ob der angegebene Wert ein oder zweistellig ist). Es genügt nur das Tagesdatum (1-31).

Nach der Bestätigung mit <return> werden Sie nun gefragt, um welche Uhrzeit Sie geweckt werden möchten. Geben Sie hier die gewünschte Uhrzeit in folgendem Format ein:

hh,mm   z.B.: 05,15

Das Format gleicht also dem obigen mit dem Unterschied, daß die Angabe der Sekunden fehlt.

Haben Sie alle dieses Angaben getätigt, beginnt der Computer mit der Arbeit. Wie er das macht, soll im folgenden erläutert werden.

Es wird also eine normale Analog-Uhr mit Stunden-, Minuten- und Sekundenzeiger sowie Datumsanzeige in hochauflösender Graphik auf den Bildschirm gezaubert. Dabei wird die aktuelle Zeit ständig aus der normalen BASIC-Uhr gewonnen. (Wie Sie wissen steht die aktuelle Zeit in der Sondervariablen TI$ in der Form hhmmss.)

Bevor der kontinuierliche Betrieb beginnt, muß zunächst einmal die Uhr gezeichnet werden. Dieser Vorgang beginnt in Zeile 310 mit dem Einschalten der Graphik. Zeile 360 schreibt eine Überschrift in die Graphik. Dabei wird ein Befehl verwendet, den wir erst im nächsten Abschnitt besprechen werden. Zerbrechen Sie sich also noch nicht den Kopf darüber. Dieser Befehl gibt einfach einen String ab einer bestimmten Koordinate auf dem Graphikbildschirm aus.

Die zwei Kreise der Uhr werden - wie Sie leicht erkennen - in den Zeilen 380/390 gezogen. Um der Uhr ein professionelles Aussehen zu geben, soll als Stundenmarke jede Stunde ein langer Strich gezeichnet werden. Dazwischen sollen noch einmal 4 kleinere Striche weiter unterteilen. Dieses Vorhaben wird in den Zeilen 430-490 in zwei verschachtelten FOR...NEXT-Schleifen realisiert:

Dabei zählen wir die Winkelvariable W von 0 bis 360 in Schritten zu 360/48 (wir müssen 48 Striche zeichnen) hoch. Zusätzlich wird die Y-Variable ständig von 0-3 gezählt. Der Wert dieser Variable ist wichtig für die Länge des Striches, der in der FOR...NEXT-Schleife der Zeilen 460-480 gezeichnet wird. Die Formel 4+4*INT(Y/3) wird genau 4, wenn Y nicht 3 ist, INT(Y/3) also Null wird. Ist Y=3, so wird der Wert der Formel 8 und ein Strich mit einer Länge von 8 Punkten wird gezeichnet. Der Strich wird durch mehrere CIRCLE-Befehle gezeichnet. Dabei wird nur an der Stelle, die der Winkel W und der Radius 62-X angeben ein Punkt gezeichnet (Start- und Endwinkel des CIRCLE-Befehls sind gleich).

Als nächstes folgen die Ziffern. Auch hier wird wieder der bisher unbekannte TEXT-Befehl verwendet. Entlang eines Kreises werden die einzelnen Ziffern ausgegeben. Beachten Sie die Konstruktion RIGHT$(X$,LEN(X$)-1) in Zeile 560. Hier wird die Zahl in X$ von

dem Leerzeichen befreit, das das Commodore-BASIC stets in die Zahlen hineinschmuggelt.

Für die Tagesanzeige wird in Zeile 610 ein Kasten vorbereitet.

Jetzt geht es richtig los. Die fixen Teile der Uhr sind gezeichnet. Es fehlen die Zeiger und das Datum:

In Zeile 690 wird zunächst getestet, ob die Weckzeit erreicht ist und entsprechend verzweigt (s.u.). In Zeile 730 wird nun aus der TI$-Variable der aktuelle Sekundenwert geholt. Hat sich der Wert im Vergleich zum bisherigen geändert (Test in Zeile 740), dann muß der Sekundenzeiger um eine Sekunde verschoben werden. Dies erledigt ein Unterprogramm ab Zeile 1120. Diesem Unterprogramm werden die notwendigen Daten übergeben (alter und neuer Sekunden-, Minuten- oder Stundenwert sowie die Länge des Zeigers in R).

Hier in Zeile 1120 wird nun die Variable ZW initialisiert und direkt nach 1150 verzweigt, falls der entsprechende Zeiger das erste Mal gezeichnet werden muß. Dann nämlich entfällt das vorherige Löschen das alten Zeigers. Da Löschen und Zeichnen durch den Invertierungsmodus geschieht, ist dieses Vorgehen notwendig. Interessant in diesem Zusammenhang sind die beiden Zeilen 1130 und 1150. Hier wird (wie oben beim Zeichnen der Teilstriche) nur ein Punkt eines Kreises angesprochen (in diesem Fall der Endpunkt eines Zeigers). Dieser Punkt wird jedoch nicht gezeichnet. Es wird vielmehr (durch Zeichenmodus 4) nur der Graphikcursor an diese Stelle bewegt, um in der jeweils nächsten Zeile dann als Startpunkt einer Linie zum Mittelpunkt zu dienen. Dieses Vorgehen erspart uns eine eigene komplizierte Kreisberechnung.

Doch wir waren beim Zeichnen des Sekundenzeigers. Dieser Vorgang wird mit Zeile 770 abgeschlossen. Hier wird der neue Sekundenwert (S2) in den Speicher für den alten Wert (S1) übertragen, um weiter oben (Zeile 740) den Vergleichswert zu haben (s.o.).

Es folgt das Zeichnen des Minutenzeigers. Hier wird nur geprüft, ob sich der Minutenwert geändert hat. Ist dies nicht der Fall, dann geht es ganz normal weiter zu den Sekunden.

Ebenso wird mit dem Stundenzeiger verfahren. Hier kommt eine Besonderheit hinzu. Anders als bei den Sekunden und Minuten tauchen hier Werte von 0 bis 23 auf. Diese Werte sind auch noch für zwei Uhrumrundungen zuständig. Weiterhin soll der Stundenzeiger nicht

von Stunde zu Stunde hüpfen, sondern gleichsam mit dem Minutenzeiger langsam von einer Stundenmarkierung zur anderen "gleiten".

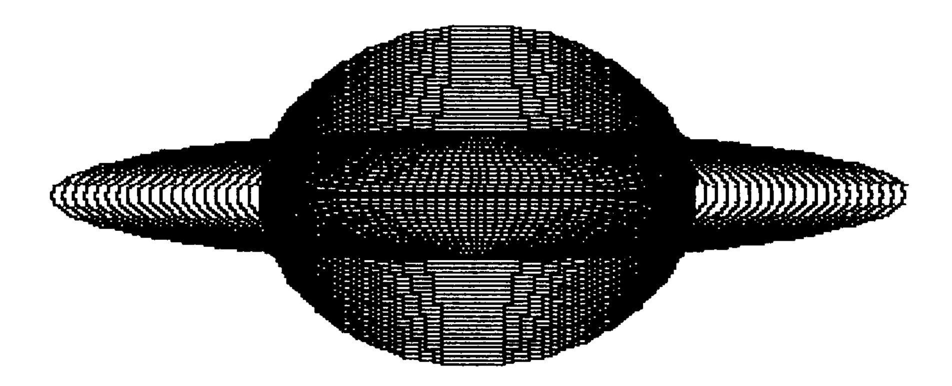
Dies wird durch die Formeln in den Zeilen 890 und 900 realisiert. In der ersten dieser beiden Zeilen wird zunächst einmal der normale Stundenwert (0-23) geholt und als Vorbereitung für die nächste Zeile durch 12 geteilt. Auf diese Weise erhalten wir bei Stundenwerten von 0-11 Werte zwischen 0 und 1, bei Stundenwerten von 12-23 Werte von zwischen 1 und 2. In der nächsten Zeile wird durch die Teilformel H2-INT(H2) die Vorkommazahl dieses Wertes abgeschnitten, wodurch sich nur noch Werte von 0-1 ergeben, die im folgenden mal 12 genommen wieder den Bereich 0-11 abdecken. Damit ereichen wir genau 1 Uhrumdrehung (statt der ursprünglichen zwei).

Nun wird in der Formel aber nicht nur mal 12, sondern mal 12 und mal 5 gleich mal 60 genommen. Dies wird so gehalten um Werte von 0-60 zu erhalten (wie Sekunden und Minutenwerte), da sonst unsere Zeigerroutine nicht richtig funktionieren würde. Zum guten Schluß wird noch der Minutenanteil hinzuaddiert. Da sich die Minuten nur auf ein zwölftel der sonstigen Strecke auswirken sollen (nämlich nur von einem Stundenteilstrich zum anderen und nicht rund um die Uhr), wird auch nur der zwölfte Teil hinzuaddiert. Dieser so entstandene Stundengesamtwert wird nun wieder genauso behandelt, wie von den Minuten und Sekunden bekannt.

War es gerade Null Uhr oder wurde das Datum noch nicht in das Kästchen gezeichnet, dann muß auch das Datum auf den Bildschirm gebracht werden, was ab Zeile 980 passiert. Auch hier wieder der TEXT-Befehl.

Der Weckton wird ab Zeile 1220 erzeugt. Da wir es hier fast nur mit den bisher noch völlig unbekannten Sound-Befehlen der Supergraphik zu tun haben, wollen wir hier nicht näher darauf eingehen. Der Rechner weckt solange, bis Sie eine beliebige Taste betätigen. Dann geht er zur normalen Anzeige über.

Scheuen Sie sich nicht, Änderungen vorzunehmen. An die Datumsanzeige kann z.B. ein immerwährender Kalender angehängt werden, die Uhrzeit könnte zusammen mit dem Datum noch einmal digital angezeigt werden. Vielleicht schließen Sie ja auch Ihre Deckenbeleuchtung, Ihr Radio und Ihre Kaffeemaschine an den Rechner an, die alle zur gleichen Zeit eingeschaltet werden etc.

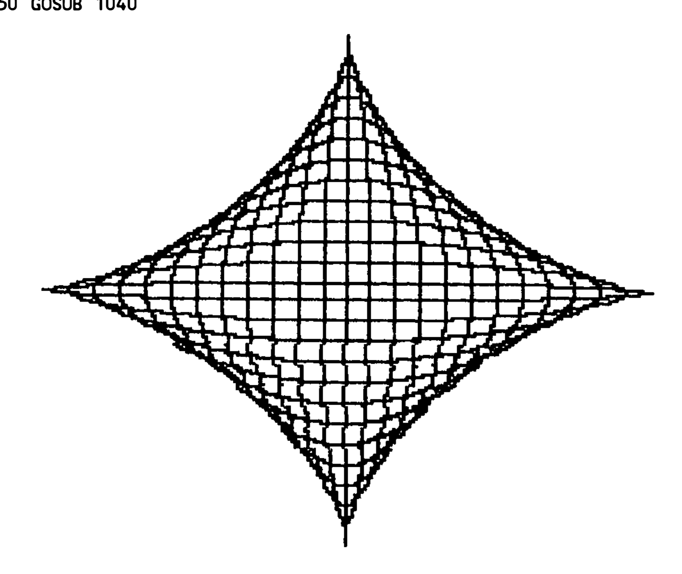


2.7.3. Einige Anwendungen

Im folgenden werden wir Ihnen wieder einige Spielereien demonstrieren, die Sie zu neuen Programmiererhöhenflügen anregen sollen.

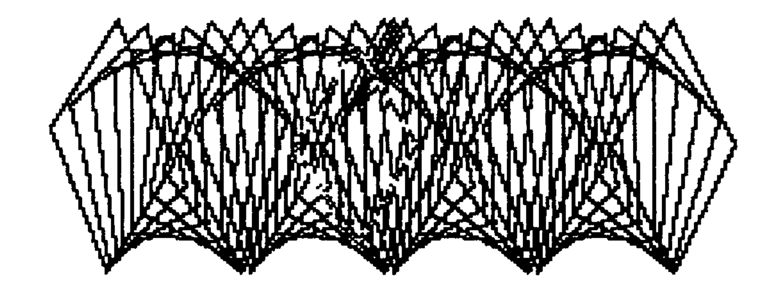
```
100 REM *********************
110 REM **                 **
120 REM **  CIRCLE-DEMOS   **
130 REM **                 **
140 REM *********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 REM
180 REM ****************
190 REM **            **
200 REM **  ALPHORN   **
210 REM **            **
220 REM ****************
230 REM
240 FOR X=1 TO 119 STEP 2
250 CIRCLE ,,1.3*X+10,50*SIN(X/30-.2)+X+10,X/2+X/8,X/2
260 NEXT X
270 PLOT ,130,105 TO 270, 50
280 PLOT ,130,110 TO 280,100
290 PLOT ,130,115 TO 270,150
300 GOSUB 1040
310 REM
320 REM ***************
330 REM **           **
340 REM **  DEMO 2   **
350 REM **           **
```

```
360 REM **************
370 REM
380 GCLEAR : COLOR= 2,2
390 FOR X=0 TO 159 STEP 4
400 CIRCLE ,,160,100,X,30
410 CIRCLE ,,160,100,80,X/2
420 NEXT X
430 GOSUB 1040
440 REM
450 REM **************
460 REM **          **
470 REM **  DEMO 3  **
480 REM **          **
490 REM **************
500 REM
510 GCLEAR : COLOR= 8,8
520 FOR X=0 TO 100 STEP 8
530 CIRCLE ,,160,100,X,100-X
540 NEXT X
550 GOSUB 1040
```



```
560 REM
570 REM **************
580 REM **          **
590 REM **  DEMO 4  **
600 REM **          **
610 REM **************
620 REM
630 GCLEAR : COLOR= 9,9
640 FOR X=1 TO 90 STEP 2
650 CIRCLE ,X/1.8,X+10,X+10,X,X,X,X+115
```

```
660 NEXT X
670 GOSUB 1040
680 REM
690 REM ***************
700 REM **           **
710 REM **  DEMO 5   **
720 REM **           **
730 REM ***************
740 REM
750 GCLEAR : COLOR= 11,11
760 FOR Z=1 TO 10
770 FOR X=1 TO 359 STEP 10
780 CIRCLE 2,90,260-X/2,100,20,50,360-X,359-X
790 CIRCLE 2,90,260-X/2,100,20,50,360-X,359-X
800 NEXT X
810 NEXT Z
820 CIRCLE 2,90,160,100,20,40,359,358
830 GOSUB 1040
840 REM
850 REM *****************
860 REM **             **
870 REM **  KLEEBLATT  **
880 REM **             **
890 REM *****************
900 REM
910 GCLEAR : COLOR= 11,11
920 FOR X=1 TO 359 STEP 2
930 A=35*SIN(X/15-.5)+40
940 CIRCLE 4,,160,100,A,A,X,X
950 PLOT , TO 160,100
960 NEXT X
970 CIRCLE ,,154,150,15,50,20,180
980 GOSUB 1040
990 END
1000 REM
1010 REM
1020 REM WARTESCHLEIFE:
1030 REM
1040 FOR T=1 TO 3000
1050 GET A$ : IF A$="" THEN NEXT T
1060 RETURN
```



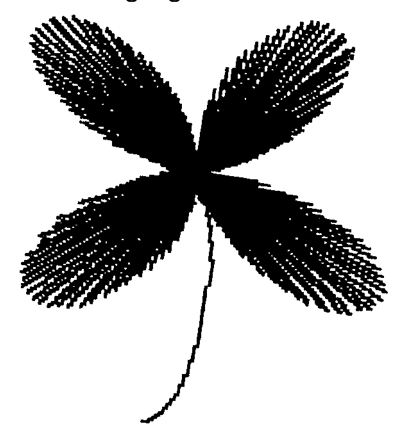
Dies ist nur ein winziger Auszug aus all den vielen Möglichkeiten, die Ihnen der CIRCLE-Befehl bietet. Wichtig auch hierbei ist, daß Sie persönlich möglichst viele Veränderungen an den Parametern usw. vornehmen, um einmal die verschiedenen Reaktionen kennenzulernen. Setzen Sie doch einmal statt Zeile 530 folgendes ein:

```
530 CIRCLE ,,160,X+100,X,100-X
```

Nur eine kleine Veränderung und schon ändert sich das ganze Bild. Vielleicht versuchen Sie auch dies:

```
530 CIRCLE ,,160,X+110,X,100-X
```

Eben nur fast identisch!
Das Prinzip ist oft einfach: die verschiedenen Parameter (Mittelpunkt, Radien, Start- und Endpunkte etc.) werden einfach auf bestimmte Kurven (oft Sinus-Kurven, aber auch Geraden wie im gerade gezeigten Beispiel) gelegt. Einige Parameter bleiben fix, andere variabel. Oft entstehen solche Figuren nur durch spielerisches Probieren, oft durch kleine mathematische Überlegungen.



2.7.4 CAD-Zeichner: 3.Teil

Wir haben einen neuen Befehl kennengelernt, also bauen wir ihn auch gleich in unseren CAD-Zeichner ein. Diesmal sind es nur ein paar wenige Zeilen, die hinzukommen, um unser Vorhaben in die Tat umzusetzen, da alles bereits vorbereitet ist. Zwei Zeilen sind zu ändern (1730, 1740) und nur 8 weitere hinzuzufügen. Laden Sie also die bisher aktuellste Version ein (s. CAD-Zeichner 2.Teil) und tippen ganz einfach die folgenden Zeilen ein:

```
510 IF A$="K" THEN FZ=4 : GOSUB 1450 :REM KREIS
1455 IF FZ=4 THEN X=10 : Y=189       :REM KREIS
1465 SL=2 : WA=0 : WE=359            :REM INITIALISIERUNG CIRCLE
1730 ON FZ GOTO 1800,1900,1900,2050  :REM SPRUNG ZU: LINIE, FRAME, FILL,
 CIRCLE
1740 ON FZ GOTO 1800,1900,2000,2050  :REM SPRUNG ZU: LINIE, FRAME, FILL,
 CIRCLE
2020 REM
2030 REM
2040 REM KREIS ZEICHNEN/LOESCHEN:
2050 CIRCLE ZM,SL,X1,Y1,X2,199-Y2,WA,WE
2060 RETURN
```

In Zeile 510 wird aus dem Hauptprogramm mit dem FLag FZ=4 in die Objekt-Routine gesprungen. Dort sorgen die beiden Zeilen 1455/1465 dafür, daß die Parameter für den Kreis korrekt initialisiert werden, da hier X und Y den Radius des Kreises steuern. Die Zeilen 1730/1740 sind ebenfalls für ein viertes Objekt modifiziert worden. Die zweitausender Zeilen schließlich zeichnen, löschen oder invertieren einen Kreis mit den entsprechenden Parametern.

Im Hauptprogramm legen Sie also mit dem normalen Graphikcursor den Mittelpunkt des Kreises oder der Ellipse fest. Nach der Taste "K" für Kreis können Sie nun mit den Cursortasten die beiden Radien verändern. Dabei sollten Sie aufpassen, daß Sie nicht in zu große oder negative Bereiche stoßen, da Sie sonst Unsinn auf dem Bildschirm erhalten. Bei f1 wird der Kreis gezeichnet und es wird zurück zum Hauptprogramm gesprungen. Bei f3 wird der Mittelpunkt beibehalten und Sie können einen zweiten Kreis zeichnen. Ansonsten bleiben alle anderen Funktionen erhalten.

Zeile 1465 ist absichtlich eingesetzt worden, um Ihnen die Möglichkeit zu schaffen, den CAD-Zeichner zu erweitern. Sie könnten beispielsweise im Objekt-Modus Start- und Endpunkt sowie die Seitenlänge des CIRCLE-Befehls mit anderen Tasten steuern. Hierzu bedarf es nur einer kleinen Erweiterung (zusätzliche Tastenabfrage und Änderung der Parameter). Vielleicht gestalten Sie ja die gesamte Kreisroutine komfortabler.

2.8 Text in hochaufgelöster Graphik

Was uns bisher stets gestört hat, ist die mangelnde Kommunikation im Graphikmodus. Das hat nun ein Ende. Von nun an werden wir auch Texte in die Graphik bringen können, an jede beliebige Stelle. Hierzu steht uns ein Befehl zur Verfügung, der uns bereits von der hochauflösenden BASIC-Uhr her bekannt sein sollte:

2.8.1 Der TEXT-Befehl

TEXT

Befehl:	TEXT zm,str$,x,y,m
Beispiel:	TEXT 0,"HALLO",100,130,0
Parameter:	zm: Zeichenmodus
	str$:zu schreibender Text
	x: x-Koordinate
	y: y-Koordinate
	m: Textmodus (0,1)
Funktion:	Setzen (Loeschen etc.) eines Textes als Graphik

Ihre Graphiken stehen von nun an nicht mehr anonym im Raum, Sie können sie jetzt beschriften oder Erklärungen abgeben. Sie beginnen bei x,y und schreiben Ihren unter "str$" abgelegten Text in die Graphikseite. Sind Sie am Ende einer Zeile, stoppt der Befehl, und Sie müssen neue Anfangskoordinaten angeben.

m ist ein Parameter, der den Textmodus auswählt:

m = 0: Groß/Graphikmodus
m = 1: Groß/Kleinschrift

Mitten im Text können Sie also Groß/Kleinschrift und Graphikzeichen miteinander gleichzeitig kombinieren. Unter Verwendung von (rvs on) und (rvs off) steht Ihnen der gesamte Zeichenvorrat des CBM 64 zur Verfügung.

SUPERGRAPHIK 64

```
100 REM ******************************
110 REM **                          **
120 REM **  TEXT, TEXT, TEXT, ...   **
130 REM **                          **
140 REM ******************************
150 REM
160 PRINT CHR$(147)  :REM BILDSCHIRM LOESCHEN
170 FRAME ,3,5,5 TO 75,45
180 TEXT ,"SUPER",20,18,0
190 TEXT ,"GRAPHIK",12,28,0
200 TEXT ,"64",31,38,0
210 WAIT 198,255
220 GMODE 0,1 : GCLEAR : SCOL= 0,0
230 FRAME 3,9,80,80 TO 230,120
240 TEXT ,"SUPERGRAPHIK 64",95,103,0
250 POKE 198,0 : WAIT 198,255
```

SUPER
GRAPHIK
64

Ein Rahmen wird gezogen und der Text "SUPERGRAPHIK 64" hineingeschrieben: In LGR, in HGR.
Versuchen Sie es doch einmal mit Ihrem Namen!

```
100 REM **************************
110 REM **                      **
120 REM **  ZUFALLSBUCHSTABEN   **
130 REM **                      **
140 REM **************************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 FOR X=1 TO 1000
180 ST$=CHR$(RND(1)*224+32)     :REM ZUFALLSBUCHSTABEN
190 TEXT ,ST$,RND(1)*320,RND(1)*200,RND(1)*2
200 NEXT X
210 WAIT 198,255
```

Wie Sie sehen, können Sie auch einzelne Buchstaben und einfache Stringspeicher für str$ einsetzen. Damit bleiben Ihnen alle Möglichkeiten offen.

2.8.2 Text nach Graphik - geht das?

Klar geht das! Wir können nicht nur einzelne Strings in die Graphik zeichnen, wir können sogar die gesamte Text-Seite in die Graphik übertragen. Nach diesem Vorgang meinen Sie, es habe sich überhaupt nichts verändert, wenn Sie aber einmal mit den normalen Graphikbefehlen in diesem Bild "herumfuschen", werden Sie sehen, was sich getan hat. Aber eins nach dem anderen:

TRANS

Befehl:	TRANS m
Beispiel:	TRANS 1
Parameter:	m: Zeichensatz (0/1)
Funktion:	Kopieren der Textseite in die Graphik

Der TRANS-Befehl ermöglicht Ihnen, Ihre gesamte Textseite mit allen Zeichen, Graphiken und Farben etc. als hochauflösende(!!!) Graphik darzustellen. Sie können also auf einfachste Art und Weise Ihre Textseite erstellen und dann in HGR (MC) weiterbearbeiten. Eine weitere Möglichkeit, die Ihnen dieser Befehl bietet und auf die wir später noch zurückkommen werden, ist die Erstellung einer Hardcopy der Textseite auf jedem Drucker. Ist das nichts?

Mit dem Parameter m geben Sie an, in welchem der zwei Zeichensätze die Übertragung erfolgen soll:

m=0 : Groß-/Graphiksatz
m=1 : Groß-/Kleinschrift

Schreiben Sie doch einfach einmal den gesamten Text-Bildschirm mit irgendwelchem Unsinn voll, verwenden Sie dabei soviele Farben wie möglich und die unterschiedlichsten Zeichen. Sind Sie mit Ihrem Werk zufrieden, dann geben Sie ein:

```
TRANS 0 : GMODE 0,1 : PLOT ,10,10 TO 300,190
```

Ihre Textseite wird in Graphik übertragen, geradezu eingefroren und quer über den Bildschirm ein dicker Strich gezogen. Wollen Sie wieder zurück, geben Sie einfach blind ein:

```
GMODE 0,0
```

Geben Sie doch einmal folgende Zeile zusätzlich in das Programm vom vorherigen Kapitel (TEXT, TEXT, TEXT, ...) ein:

```
225 TRANS 0
```

2.8.3 Vergrößerung, Verzerrung, Drehung

Wir können bereits schöne Buchstaben auf den Bildschirm zeichnen. Doch mit relativ einfachen Mitteln sind wir in der Lage, diese Buchstaben mathematisch zu "bearbeiten". Sehen wir uns doch einmal ein paar Programme an und besprechen dann die einzelnen Effekte:

a.) Vergrößerung:

```
100 REM **********************
110 REM **                  **
120 REM **  VERGROESSERUNG  **
130 REM **                  **
140 REM **********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 TEXT ,"&",0,7,0          :REM VERGROESSERUNGSOBJEKT
180 X1=10 : Y1=40            :REM STARTKOORDINATEN
190 FOR V=.5 TO 5.5 STEP .5 :REM VERGROESSERUNGSFAKTOR
200   X1=X1+V*8 : Y1=Y1      :REM AKTUELLE STARTKOORDINATEN
```

```
210   FOR X0=0 TO 7 STEP .99/V
220     FOR Y0=0 TO 7 STEP .99/V
230       PLOT T TEST,4,X0,Y0
240       IF TEST=1 THEN : PLOT ,X1+X0*V,Y1+Y0*V : GOTO 260
250       REM PLOT 1,X1+X0*V,Y1+Y0*V
260     NEXT Y0
270   NEXT X0
280 NEXT V
290 REM
300 REM ************************
310 REM **                    **
320 REM **  VERGROESSERUNG 2  **
330 REM **                    **
340 REM ************************
350 REM
360 X1=10 : Y1=120        :REM STARTKOORDINATEN
370 V=1                   :REM STARTVERGROESSERUNGSFAKTOR
380 FOR T=1 TO 11         :REM ANZAHL BUCHSTABEN
390   X1=X1+V*9 : Y1=Y1   :REM AKTUELLE STARTKOORDINATEN
400   FOR X0=0 TO 7 STEP 1/(2*V)
410     V=V+.01
420     FOR Y0=0 TO 7 STEP .9/V
430       PLOT T TEST,4,X0,Y0
440       IF TEST=1 THEN : PLOT ,X1+X0*V,Y1+Y0*V : GOTO 460
450       REM PLOT 1,X1+X0*V,Y1+Y0*V
460     NEXT Y0
470   NEXT X0
480 NEXT T
490 WAIT 198,255
```

Nur die Ruhe, wird alles erklärt! Was haben wir gemacht? Nun, im ersten Teil dieses Programmes haben wir ein Zeichen (in diesem Fall das kaufmännische "und", Sie können jedoch jedes beliebige Zeichen einsetzen) kontinuierlich vergrößert. Dabei diente uns ein normales Zeichen links oben in der Ecke quasi als Vorlage.

Zunächst einmal wird das betreffende Zeichen eben in besagte Ecke geschrieben (Zeile 170). Dann werden drei ineinander verschachtelte FOR...NEXT-Schleifen eröffnet. Die äußere mit der Laufvariablen V ändert bei jedem Durchlauf den Vergrößerungsfaktor V um 0,5 (Zeile 190). Die Koordinaten, an denen das vergrößerte Zeichen starten soll (angegeben wird stets der obere linke Punkt des Zeichens) wurden bereits in Zeile 180 initialisiert und werden nun in Zeile 200 soweit verändert, daß jedes Zeichen genügend Platz hat, egal wie groß es wird.

&

* & & & & & & & & & & &

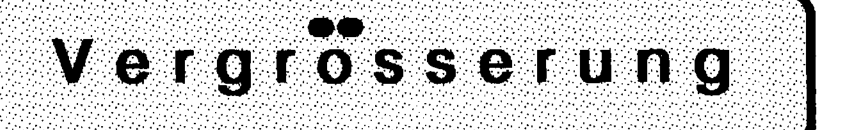
Interessant sind nun die beiden folgenden FOR...NEXT-Schleifen. Hier wird jeder Punkt des Vorlagenzeichens (8x8-Matrix) einzeln abgefragt. Dabei erhöht die äußere Schleife ständig die Spalte X0, die innere die Zeile Y0. Wenn Sie kritisch sind, werden Sie fragen, wieso dabei Zeile und Spalte nicht jedesmal um 1 erhöht werden. Das wäre rein theoretisch möglich. In diesem Fall aber würde zwar das Zeichen vergrößert, es hätte aber genau so viele Punkte (64) wie das Originalzeichen. Die einzelnen Punkte lägen dann also nur weiter auseinander. Das kann ja nicht unser Ziel sein.

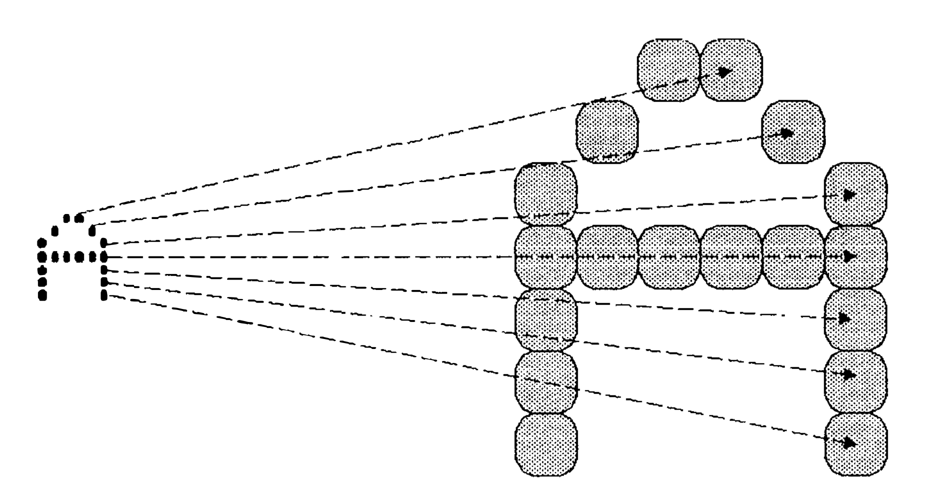
Tatsächlich abgefragt wird jeder einzelne der 8x8=64 Punkte unseres Zeichens in Zeile 230. Hier wird die Variable TEST gleich 0, wenn an der betreffenden Stelle kein und gleich 1, wenn dort ein Punkt gesetzt ist. Ist ein Punkt gesetzt, so muß natürlich an der entsprechenden Stelle des vergrößerten Zeichens ebenfalls ein Punkt gesetzt werden. Dies geschieht in Zeile 240. Hier findet auch die Vergrößerung statt. Die Zeichenkoordinaten X0 und Y0 werden mit dem Vergrößerungsfaktor malgenommen und zu der Startkoordinate (X1/Y1) hinzuaddiert.

Ist der Punkt des Zeichens nicht gesetzt, so erfolgt in der abgedruckten Version nichts. Wenn Sie aber das "REM" aus Zeile 250 entfernen, so wird an der entsprechenden Stelle ein Punkt gelöscht. Das kann in manchen Fällen wichtig sein, wenn es auch in unserem Beispiel keinen Unterschied bewirkt.

Das Prinzip ist also klar: Man schreibt (vielleicht später unsichtbar in schwarz) ein Zeichen in den Graphikbildschirm und tastet mit der Test-Funktion des PLOT-Befehls jeden einzelnen Punkt der Zeichenmatrix ab. Mit den gewonnenen Koordinaten der gesetzten Punkte des Zeichens können wir nun machen, was wir wollen: Wir können vergrößern, verzerren, spiegeln, drehen, projizieren und so weiter.

Im zweiten Programmteil wird nicht jedes Zeichen mit einem für das Zeichen konstanten Faktor vergrößert, der Vergrößerungsfaktor nimmt vielmehr auch innerhalb des Zeichens zu. Ein Zeichen wird also hinten größer also vorne. Man erreicht hierdurch eine gleitende Vergrößerung.





b.) Spiegelung:

```
100 REM ******************
110 REM **              **
120 REM **  SPIEGELUNG  **
130 REM **              **
140 REM ******************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 TEXT ,"R",0,7,0       :REM SPIEGELOBJEKT
180 X1=160 : Y1=100       :REM STARTKOORDINATEN
190 X2=150 : Y2= 90       :REM STARTKOORDINATEN
200 XM=(X2+X1)/2
210 YM=(Y2+Y1)/2
```

```
220 PLOT ,XM, 0 TO XM,199
230 PLOT , 0,YM TO 319,YM :REM SPIEGELACHSEN
240 REM
250 FOR X0=0 TO 7
260   FOR Y0=0 TO 7
270     PLOT T TEST,4,X0,Y0
280     IF TEST=0 THEN 330
290     PLOT ,X1+X0,Y1+Y0 :REM NORMAL
300     PLOT ,X1+X0,Y2-Y0 :REM X-GESPIEGELT
310     PLOT ,X2-X0,Y1+Y0 :REM Y-GESPIEGELT
320     PLOT ,X2-X0,Y2-Y0 :REM X + Y-GESPIEGELT
330   NEXT Y0
340 NEXT X0
350 WAIT 198,255
360 REM
370 REM ********************
380 REM **                **
390 REM **  SPIEGELUNG 2  **
400 REM **                **
410 REM ********************
420 REM
430 GCLEAR : SCOL= 0,0
440 TEXT ,"SPIEGELUNG",20,10,0
450 CIRCLE ,,50,70,40,30
460 PLOT , 5,25 TO 140,60
470 FRAME 3,4,120,70 TO 158,98
480 REM
490 XA=160 : YA=100       :REM SPIEGELACHSENNULLPUNKT
500 PLOT ,XA, 0 TO XA,199
510 PLOT , 0,YA TO 319,YA :REM SPIEGELACHSEN
520 REM
530 FOR X0=0 TO XA-1      :REM FELD ABTASTEN
540   FOR Y0=0 TO YA-1
550     PLOT T TEST,2,X0,Y0
560     PLOT 2,X0,Y0
570     IF TEST=0 THEN 610
580     PLOT ,     X0,2*YA-Y0 :REM X-GESPIEGELT
590     PLOT ,2*XA-X0,     Y0 :REM Y-GESPIEGELT
600     PLOT ,2*XA-X0,2*YA-Y0 :REM X + Y-GESPIEGELT
610   NEXT Y0
620 NEXT X0
630 POKE 198,0 : WAIT 198,255
```

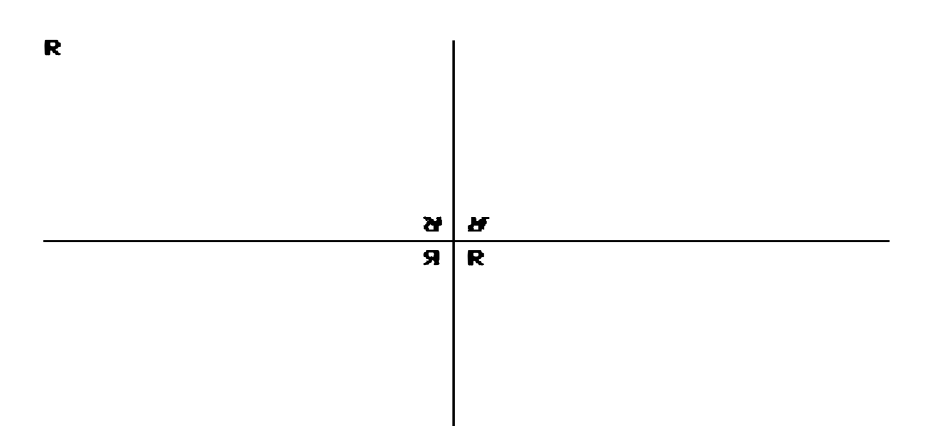
Interessante Dinge lassen sich durch Spiegelungen erreichen. Das abgedruckte Programm zeigt Ihnen, wie Sie dabei vorgehen können.

Im ersten Programmteil beschränken wir uns zunächst einmal auf die Spiegelung eines einzigen Buchstabens. Dabei wenden wir die gleiche Technik an, wie wir sie bereits beim Vergrößern von Buchstaben kennengelernt haben: Wir schreiben einen Buchstaben (hier "R") in die obere linke Ecke des Bildschirms (Zeile 170) und tasten ihn Punkt für Punkt ab (Zeile 270). Da wir diesen Buchstaben jeweils um die Waagerechte, die Senkrechte und um beide gleichzeitig spiegeln möchten, legen wir zunächst einmal die Koordinaten fest, bei denen die gespiegelten Buchstaben erscheinen sollen (Zeilen 180/190). Dabei wird der originale Buchstabe nach (160/100) kopiert. Um Ihnen die Spiegelung etwas zu verdeutlichen, werden in den Zeilen 220 und 230 noch kurz die Spiegelachsen eingezeichnet.

Dann geht es richtig los. Wie bekannt werden alle Punkte des Zeichens abgetastet. Die gesetzten Punkte werden dann ab Zeile 290

- original nach X1,Y1 kopiert
- um die x-Achse gespiegelt
- um die y-Achse gespiegelt
- um beide Achsen gespiegelt

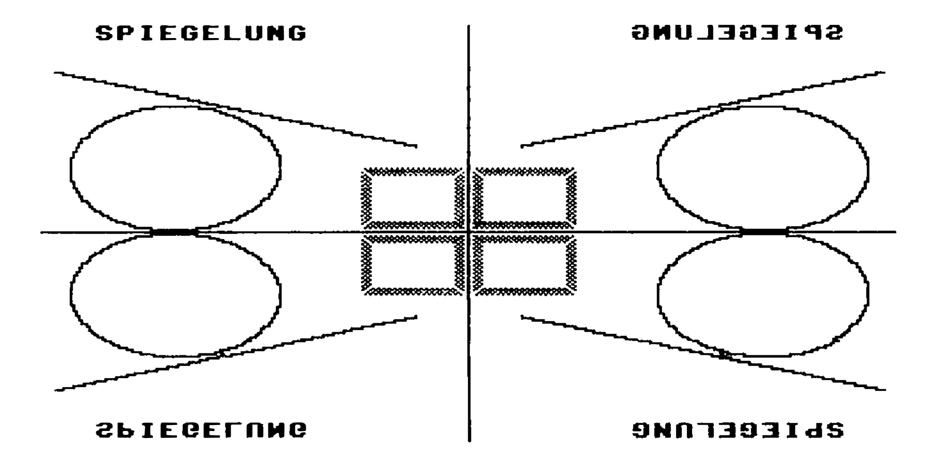
Dies geschieht durch Addition bzw. Subtraktion der einzelnen Koordinaten von den Startkoordinaten. Wenn Sie sich die Prozedur anschauen, werden Sie erstaunt sein, wie mathematisch simpel eine solche Spiegelung ist.



Im zweiten Teil wird die Spiegelung nun auf einen ganzen Bildschirmbereich ausgedehnt. Der Bildschirm wird durch zwei Spiegelachsen in 4 gleich große Teile geteilt (Zeilen 500 und 510). Das obere linke Viertel dient dabei als Vorlage und wird jeweils um die Achsen in die drei anderen Viertel gespiegelt.

Dazu werden zunächst einmal diverse Objekte in besagten Bildschirmbereich eingezeichnet (Zeilen 440-470). Dann wird das gesamte Feld in zwei FOR...NEXT-Schleifen (Zeilen 530-620) Punkt für Punkt abgetastet (wie sonst die Zeichenmatrix). Für jeden gesetzten Punkt innerhalb dieses Feldes wird in den anderen drei Feldern jeweils an der richtigen Stelle ebenfalls ein Punkt gesetzt. Sie müssen sich etwas in die Formeln der Zeilen 580-600 hineindenken. Sie werden sehen, es ist recht einfach. Um Ihnen die Sache nicht zu langweilig werden zu lassen, wird gleichzeitig mit dem Punkt-Test in Zeile 550 ein Punkt invertiert gezeichnet, der natürlich in der folgenden Zeile wieder gelöscht werden muß. Auf diese Weise können Sie stets den kleinen abtastenden Punkt beobachten, der auf dem Original hin- und herläuft.

Die beiden gerade besprochenen Programmteile zur Spiegelung sollten Ihnen als Vorlage für Ihre eigenen Spiegeleien dienen. Gerade die Spiegelung zeigt mit geringem Arbeits- und Zeitaufwand sehr schöne Effekte.




Spiegelung

c.) Drehung:

```
100 REM ***************
110 REM **           **
120 REM **  DREHUNG  **
130 REM **           **
140 REM ***************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 TEXT ,"R",0,7,0          :REM DREHOBJEKT
180 X1=10 : Y1=40            :REM STARTKOORDINATEN
190 V=1.5                    :REM VERGROESSERUNGSFAKTOR
200 PI=3.14159265
210 FOR D=0 TO PI STEP 22.5*PI/180
220   X1=X1+30
230   FOR X0=0 TO 7 STEP 1/(V+.5)
240     FOR Y0=0 TO 7 STEP 1/(V+.5)
250       PLOT T TEST,4,X0,Y0
260       IF TEST=1 THEN : PLOT ,X1+V*(X0*COS(D)-
Y0*SIN(D)),Y1+V*(X0*SIN(D)+Y0*COS(D))
270     NEXT Y0
280   NEXT X0
290 NEXT D
```

Bisher haben wir die einzelnen Zeichen stets aufrecht und wohlgeformt vor uns auf dem Bildschirm erscheinen sehen (von der Spiegelung einmal abgesehen). Wir können natürlich gleichfalls jedes beliebige Objekt in alle Richtungen drehen. Daß wir uns in diesem Abschnitt, wie auch bei den vorherigen meist auf Buchstaben versteifen, soll nicht heißen daß z.B. die Drehung oder Vergrößerung nicht auch auf Linien, Kreise, Ellipsen, Häuser oder ganze Bilder übertragbar wären, wie wir dies bereits bei der Spiegelung demonstriert haben. Doch nichtsdestotrotz bleiben wir auch bei diesem Beispiel bei unserem Lieblingsobjekt, den Buchstaben.

R



Das obige Programm zeichnet eine Reihe von "R" jeweils um 22,5 Grad verdreht auf den Bildschirm. Die Abtastroutine in den Zeilen 230-280 sollte Ihnen inzwischen geläufig sein. Lediglich die Formel in Zeile 260, die verwendet wird, um die abgetasteten Punkte gedreht

wieder aufzuzeichnen ist Ihnen bisher unbekannt. Das ganze wird Ihnen plausibler, wenn Sie die Formeln zum Drehen eines Punktes um den Winkel D mit Drehzentrum im Koordinatennullpunkt sehen:

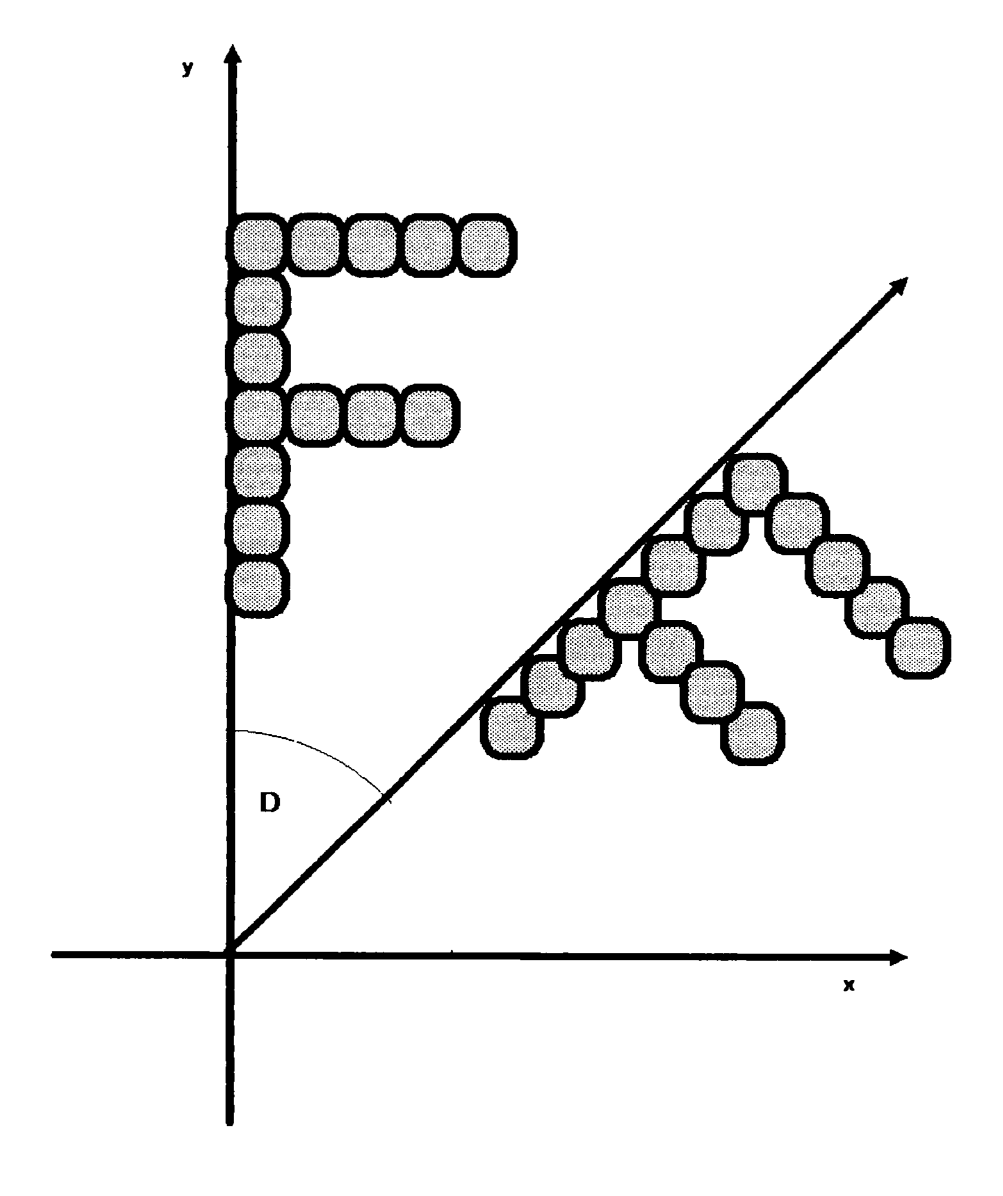
x'= x*cos(D)-y*sin(D)
y'= x*sin(D)-y*cos(D)

Dabei bedeuten:

x,y: Die Koordinaten des nicht gedrehten Punktes
x',y': Die Koordinaten des gedrehten Punktes
D: Der Drehwinkel um den Nullpunkt

Genau diese Formeln finden Sie in der betreffenden Zeile angewandt. Dort werden die Werte noch mit einem Vergrößerungsfaktor V malgenommen und zu den Startkoordinaten X1,Y1 hinzuaddiert. Sie werden sehen, daß der Prozess des Zeichnens bereits wesentlich länger dauert, als wir es bisher gewohnt waren.

Drehung



2.8.4 Text im Raum

Zum Abschluß unserer Buchstabenmanipulationen möchten wir Ihnen noch eine recht schöne Anwendung vorstellen. Dabei wird ein ganzer Schriftzug auf eine Kugel projiziert. Den Schriftzug können Sie selbstverständlich durch Ihren eigenen Namen ersetzen. Bitte haben Sie etwas Geduld. Das Zeichnen dieses Bildes dauert einige Zeit. Machen Sie vielleicht in der Zwischenzeit einige Besorgungen, Ihr Rechner wird ein paar Stunden beschäftigt sein. Die Anregung zu diesem Programm hat der Autor dem wirklich hervorragenden und für Leute mit Sinn fürs Schöne empfehlenswerten Buch "Spiele mit Computergrafik" von Norbert Teitz (Hagemann-Verlag) entnommen.

```
100 REM ***********************
110 REM **                   **
120 REM **  KUGELPROJEKTION  **
130 REM **                   **
140 REM ***********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 SI=SIN(1) : CO=COS(1)
180 S$="SUPERGRAPHIK-64/"
190 BO=-1
200 FILL 1,0,0 TO 7,7             :REM BUCHSTABEN LOESCHEN
210 A=A+1 : IF A>LEN(S$) THEN A=1 :REM NAECHSTER BUCHSTABE
220 TEXT ,MID$(S$,A,1),0,7,0      :REM AUSGEBEN
230 BO=BO+0.008/COS(BO)           :REM WINKELWERTE FUER KUGEL
240 LO=LO+0.2  /COS(BO)
250 IF BO>1.3 THEN 420
260 FOR XO=0 TO 7 STEP 0.5
270   FOR YO=0 TO 7 STEP 0.5
280     PLOT T TEST,4,XO,7-YO
290     IF TEST=0 THEN 390
300     B=BO+0.02*YO
310     L=LO+0.02/COS(BO)*XO
320     X=SIN(L)*COS(B)
330     Y=COS(L)*COS(B)
340     Z=SIN(B)
350     X1= Y*CO+Z*SI
360     Y1=-Y*SI+Z*CO
370     IF X1<0 THEN 200          :REM UNSICHTBAR
380     PLOT ,160+100*X,100-100*Y1
390   NEXT YO
400 NEXT XO
410 GOTO 200
420 WAIT 198,255
```

2.8.5 CAD-Zeichner: 4.Teil

Natürlich darf ein so wichtiger Befehl wie TEXT nicht in unserem CAD-Zeichner fehlen. Wir begnügen uns hier mit der reinen Implementierung des Befehls. Sie können natürlich nach Lust und Laune Gebrauch von Spiegelung, Drehung, Vergrößerung machen, wie es sich für ein anspruchsvolles CAD-Programm gehört.

Wie immer geben wir hier nur die Zeilen an, die neu hinzukommen bzw. die sich ändern. Sie werden sich wahrscheinlich schon denken können, daß es sich hierbei wieder um recht wenige Zusätze handelt, da prinzipiell bereits alles für eine Erweiterung vorbereitet ist. Doch schauen Sie selbst.

```
520 IF A$="T" THEN FZ=5 : T$="" : GOSUB 1450 : REM TEXT SCHREIBEN
1640 IF (A>=32 AND A<=127) OR A>=160 THEN IF LEN(T$)<=40 THEN T$=T$+A$ :
REM TEXT
1730 ON FZ GOTO 1800,1900,1900,2050,2100 :REM SPRUNG ZU: LINIE, FRAME, F
ILL, ...
1740 ON FZ GOTO 1800,1900,2000,2050,2100 :REM SPRUNG ZU: LINIE, FRAME, F
ILL, ...
2070 REM
2080 REM
1090 REM TEXT ZEICHNEN/LOESCHEN:
2100 IF T$="" THEN:PLOT ZM,X2,Y2 : GOTO 2120
2110 TEXT ZM,T$,X2,Y2,1
2120 RETURN
```

Viel braucht hier nicht gesagt werden, wenn Sie bereits die früheren Versionen dieses Programmes aufmerksam beobachtet haben. Durch Zeile 520 wird Ihnen im Hauptprogramm durch Druck auf die Taste "T" ermöglicht, normalen Text mit Groß- und Kleinbuchstaben in Ihre Graphik einzugeben. Dabei sind alle Zeichen im Groß-/Klein-Modus zugelassen (Zeile 1640). Der Text wird in T$ gespeichert und in Zeile 2110 ausgegeben. Mit den Cursortasten können Sie nun die Position des gesamten Strings auf dem Bildschirm hin- und herbewegen.

Leider können Sie bereits eingetippte Buchstaben nicht mehr editieren. Haben Sie sich vertippt, dann hilft nur noch ein Betätigen der DEL-Taste und der gesamte String ist gelöscht. Hier könnten Sie durch ein paar wenige zusätzliche Befehle Abhilfe schaffen.

Die Eingabe von <rvs on> und <rvs off> zur Invertierung der Buchstaben wäre eigentlich möglich, wurde hier aber nicht erlaubt. Das können Sie nachholen. Dazu müssen Sie den IF-Ausdruck in Zeile 1640 erweitern.

Ich bin sicher, die Erweiterung durch den TEXT-Befehl wird Ihnen sehr viel Freude bereiten.

2.9 Shapes, schon 'mal gehört?

Shapes sind selbständige kleine Graphiken, die *softwaremäßig* unabhängig von der übrigen Graphik auf den Bildschirm gezeichnet werden. Sie können auf dem Bildschirm bewegt und eventuell sogar vergrößert und gedreht werden. Es gibt zwei verschiedene Arten von Shapes. Bei der einen wird das gesamte Shape in einer n mal m Punktematrix untergebracht - etwa vergleichbar mit einer Buchstabenmatrix in HGR -, die andere - in der Supergraphik verwirklichte - Möglichkeit bedient sich sogenannter *Zeichenvektoren.* Die besagte Figur wird definiert durch eine Reihe von Vektoren und Punktbefehlen, die jeweils einen Graphikcursor bewegen und an definierten Stellen Punkte setzen.

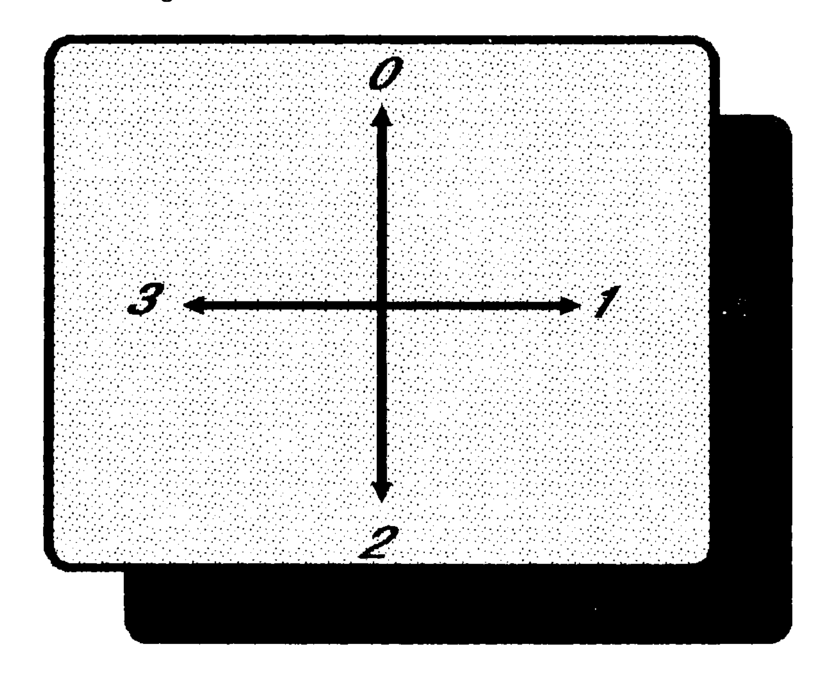
Das Ganze sieht also so aus: Sie geben mit dem Befehl zum Zeichnen eines Shapes die Startkoordinaten an, an die der Graphikcursor gesetzt wird. Von diesem Punkt ausgehend können Sie nun den Graphikcursor einen Punkt bzw. eine Einheit nach rechts, links, oben oder unten bewegen. Soll an dieser Stelle ein Punkt gesetzt werden, so geben Sie dies in dieser Definition an. Im Shape werden dann alle direkt neben- oder übereinander liegenden Punkte miteinander verbunden. So können Sie ein recht komplexes Gebilde konstruieren, das Sie zudem noch vergrößern und drehen können. Aber sehen Sie selbst:

DRAW

Befehl:	DRAW zm,str$ ON x,y
Beispiel:	DRAW 2,A$ ON 160,100
Parameter:	zm: Zeichenmodus
	str$:Definitionsstring
	x: x-Koordinate
	y: y-Koordinate
Funktion:	Setzen (Loeschen etc.) einer frei definierbaren Figur

Überlegen Sie sich, welche Möglichkeiten Ihnen dieser Befehl gibt! Sie können eine Figur beliebiger Ausdehnung frei definieren (in Form des Strings str$) und jederzeit an der gewählten x,y-Koordinate auf dem Bildschirm erscheinen lassen, weiterbewegen, vergrößern und rotieren lassen. Bei vielen - auch recht teuren - Computern ist dies die einzige Möglichkeit, bewegliche Figuren in z.B. Spielen etc. einzusetzen. Vorteil dieser Figurendefinition ist die Möglichkeit, immer wieder auftauchende Figuren einfach und bequem einmal zu definieren und immer wieder zu zeichnen.

Wie macht man das? Nun, zunächst einmal muß die Figur definiert werden. Dies geschieht sehr einfach durch Bewegen des unsichtbaren Graphikcursors um jeweils einen Punkt (eine Einheit) in eine der vier Grundrichtungen nach oben, unten, rechts, links. Jeder Richtung ist eine Ziffer zugeordnet:



0 = hoch
1 = rechts
2 = runter
3 = links

Wollen Sie nun vom Ausgangspunkt z.B. nach oben, so schreiben Sie eine 0 in den zuständigen String. Auf dem Bildschirm jedoch wird sich solange nichts tun, bis die Aufforderung kommt, einen Punkt an der derzeitigen Position des Graphikcursors zu zeichnen. Dies geschieht mit dem Zeichen "." (Punkt). Gehen Sie danach mit dem Graphikcursor wieder weiter und setzen sofort wieder einen Punkt, so werden diese beiden Punkt miteinander verbunden. Gehen Sie dagegen zwei oder mehr Schritte weiter, ohne einen Punkt zu zeichnen, dann werden die beiden nächst zusammenliegenden Punkte nicht miteinander verbunden. Das Beispiel eines Rechtecks möge dies erläutern (zur besseren Beobachtung im LGR-Modus):

```
10 GMODE 0,0 : PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
20 SCALE 0,0
30 DRAW ,".0.0.0.0.1.1.1.1.1.1.2.2.2.2.3.3.3.3.3.3." ON 39,24
```

Wie Sie sehen, wurde noch ein anderer Befehl (SCALE) verwendet; er dient zur Vergrößerung und Rotation der Figur um den Anfangspunkt (s.u.). Sollte Ihre Figur einmal über den Rand hinausragen, so kommt sie zweckmäßiger Weise gegenüber wieder zum Vorschein.

Angenommen, Sie beabsichtigen, ein vorgegebenes Gebilde (z.B. einen Buchstaben) mit Hilfe des DRAW-Befehls auf den Bildschirm zu bringen. In diesem Fall gehen Sie folgendermaßen vor. Zeichnen Sie das gewünschte Gebilde eins zu eins auf ein Blatt mit normalen Rechenkästchen. Dabei symbolisieren die Schnittpunkte zweier Linien des Rechenblatt-Rasters einen Punkt auf dem Graphikbildschirm.

Suchen Sie sich nun einen geeigneten Startpunkt, um die Figur in möglichst wenigen Anweisungen zu zeichnen. Bewegen Sie nun, wie in der unten stehenden Skizze demonstriert, Ihren Bleistift von Punkt zu Punkt der Figur und zeichnen einen Punkt an die Stellen, an denen ein Punkt auf dem Bildschirm erscheinen soll. Beachten Sie, daß Sie immer nur in die vier Grundrichtungen auf einmal gehen können, nie diagonal!

Haben Sie die Figur insgesamt abgeschritten, dann übersetzen Sie die Gehrichtungen Ihres Bleistifts in das obige DRAW-Format und schon haben Sie die notwendige Definition, die in einem String untergebracht wird. Die folgenden Programme werden Ihnen zeigen, wie Sie den DRAW-Befehl einsetzen können. Doch bevor wir uns diesen Beispielen zuwenden, sollten hier noch einige andere Möglichkeiten des Shape-Einsatzes vorgestellt werden.

DRAW

Befehl:	DRAW zm,str$
Beispiel:	DRAW 2,A$
Parameter:	zm: Zeichenmodus str$:Definitionsstring
Funktion:	Setzen (Loeschen etc.) einer frei definierbaren Figur vom Graphikcursor aus

Dieser Befehl funktioniert analog zu dem gerade besprochenen. Hier wird die Figur jedoch vom unsichtbaren Graphikcursor aus gezeichnet. Dies ist z.B. von besonderem Vorteil, wenn Ihre Figur so groß ist, daß ein String als Definition nicht mehr ausreicht. So können nun beliebig viele Teilstücke hinzugefügt werden, um ganz komplexe Figuren zu zeichnen.

```
10 GMODE 0,0 : PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
20 SCALE= 0,0
30 DRAW ,".0.0.0.0.1.1.1.1.1.1." ON 39,24
40 DRAW ,".2.2.2.23.3.3.3.3.3."
```

An die erste Hälfte des Rechtecks wird die 2. angehängt. Dieser Befehl arbeitet wie PLOT TO ... mit dem unsichtbaren Graphikcursor. Sie sollten also bereits aus dem Linienkapitel wissen, was unter einem solchen Graphikcursor zu verstehen ist.

SCALE=

Befehl:	SCALE= r,s
Beispiel:	SCALE= 1,5
Parameter:	r: Rotationswinkel
	s: Größe
Funktion:	Rotieren und Vergrößern einer DRAW-Figur

Dieser Befehl wird in Zusammenhang mit DRAW verwendet:
Mit r bestimmen Sie den Rotationswinkel Ihrer DRAW-Figur. Diese Figur wird dann um Ihren Ausgangspunkt (also um den Startpunkt des DRAW-Befehls) gedreht gezeichnet, wobei r Werte von 0-63 annehmen kann, um eine volle Umdrehung zu erreichen. Eine Einheit entspricht dabei 5,625 (5 5/8) Altgrad. Die Funktion dieses Befehls ist allerdings von dem Vergrößerungsfaktor s abhängig: Bei s=0 beispielsweise gibt es nur 8 Drehrichtungen, d.h. r=1 hat den gleichen Effekt wie r=0.

Mit s wird der Vergrößerungsfaktor s+1 einer DRAW-Figur bestimmt, wobei s Werte von 0-255 annehmen kann. Bei sehr großen Vergrößerungsfaktoren ragen Teile Ihrer Figur über die Ränder hinaus und kommen auf der anderen Seite wieder zum Vorschein (s. DRAW), was dann manchmal zu einem wahren Strichchaos führen kann.

```
10 GMODE 0,1 : GCLEAR
20 SCALE= 16,5
30 DRAW ,".3.3.3.3.0.0.0.0.0.0." ON 159,99
40 WAIT 198,255
```

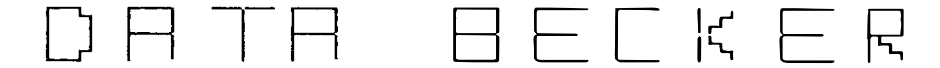
Von der Mitte des Bildschirms aus wird ein L um 16*5,625 = 90 Grad gedreht und 5+1=6 fach vergrößert gezeichnet.

Mit diesen Befehlen werden Sie viel Spaß bekommen. Versuchen Sie dies doch einmal:

```
10 GMODE 0,1 : GCLEAR
20 A$=".3.3.0.0.0."
30 FOR X=0 TO 64
40 SCALE= x,10   : DRAW 1,A$ ON 159,99
50 SCALE= x+1,10 : DRAW 0,A$ ON 159,99
60 NEXT X
70 WAIT 198,255
```

Setzen Sie doch einmal andere Werte für s ein oder tauschen Sie s und r aus. Es gibt unzählbar viele Möglichkeiten.

Auch die beiden folgenden Programme sollen Ihre Kreativität anregen. Probieren Sie doch einmal:



P R A E S E N T I E R T

```
100 REM ***************
110 REM **           **
120 REM **  SHAPES   **
130 REM **           **
140 REM ***************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 X=160 : Y=100       :REM SHAPEPOSITION
180 S= 10 : R=  0       :REM GROESSE+ROTATION INIT
190 DIM S$(9)
200 N=0 : FL=0
210 REM
220 REM HAUPTSCHLEIFE:
230 REM
240  SCALE= R,S          :REM GROESSE/ROTAITON SETZEN
250  GOSUB 610           :REM SHAPE ZEICHNEN
260  GET A$ : IF A$="" THEN 260
270  A=ASC(A$)
280  GOSUB 610           :REM SHAPE LOESCHEN
290  REM
300  REM TASTATURAUSWERTUNG:
310  REM
320  IF A$="Q" THEN END
330  IF A$="W" THEN FL=ABS(FL-1) :REM FLAG WECHSELN
340  IF FL=0 THEN 500
350  REM
360  REM BEI FLAG=1:
370  REM
380  IF A= 29 THEN S=S+1 :REM GROESSE
390  IF A=157 THEN S=S-1
400  IF A= 17 THEN R=R+1 :REM ROTATION
410  IF A=145 THEN R=R-1
420  IF S>255 THEN S=0   :REM WERTE UEBERPRUEFEN
430  IF S<  0 THEN S=255
```

```
440  IF R>255 THEN R=0
450  IF R<  0 THEN R=255
460 GOTO 240
470  REM
480  REM BEI FLAG=0:
490  REM
500  IF A = 29 THEN S$(N)=S$(N)+"1" :REM RECHTS
510  IF A =157 THEN S$(N)=S$(N)+"3" :REM LINKS
520  IF A = 17 THEN S$(N)=S$(N)+"2" :REM RUNTER
530  IF A =145 THEN S$(N)=S$(N)+"0" :REM HOCH
540  IF A$=" " THEN S$(N)=S$(N)+"." :REM PLOT
550  IF LEN(S$(N))=255 THEN N=N+1   :REM NAECHSTEN DEFINITIONSSTRING
560 GOTO 240
570 REM
580 REM
590 REM SHAPE ZEICHNEN/LOESCHEN:
600 REM
610 DRAW 2,S$(0) ON X,Y :REM ERSTEN STRING ZEICHNEN
620 FOR T=1 TO 9
630   DRAW 2,S$(T)      :REM EVENTUELL FOLGENDE STRINGS ANHAENGEN
640 NEXT T
650 RETURN
```

Dieses erste der beiden vorzustellenden Programme stellt ein kleines, ausbaufähiges Programm zur einfachen Konstruktion von Shapes bzw. Shapedefinitionen dar. Mit Hilfe dieses Programmes können Sie direkt auf dem Graphikbildschirm in der gewünschten Vergrößerung und mit dem von Ihnen zu bestimmenden Rotationswinkel Shapes (fast) beliebiger Länge konstruieren. Die Definition dieser Shapes wird sofort in einem String-Array S$(n) festgehalten.

Folgende Funktionen stehen Ihnen in diesem Programm zur Verfügung:

Q - Programmende
W - Wechsel zwischen Modus 0 und 1

unter Modus 0:

CRSR hoch: Graphikcursor eine Einheit hoch
CRSR runter: Graphikcursor eine Einheit hinunter
CRSR rechts: Graphikcursor eine Einheit rechts
CRSR links: Graphikcursor eine Einheit links
SPACE: Punkt setzen und mit evtl. vorigem Punkt verbinden

unter Modus 1:

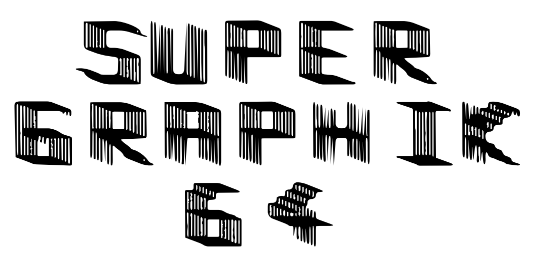
CRSR hoch: Rotationswinkel verringern
CRSR runter: Rotationswinkel vergrößern
CRSR rechts: Vergrößern
CRSR links: Verkleinern

In kurzen Worten soll hier das Prinzip dieses Programmes erläutert werden: Nach der Initialisierung verschiedener Werte (Startkoordinaten, Startgröße, Startrotation etc.) beginnt in Zeile 240 die ständig durchlaufene Hauptschleife. Hier wird die neue (bzw. aktuelle) Größe und Rotation des Objektes realisiert (SCALE=-Befehl) und das neue Shape an den aktuellen Koordinaten gezeichnet (Zeile 250). Hierzu wird ein Unterprogramm aufgerufen, das in Zeile 610 beginnt und lediglich das Shape mit der im String-Array S$(n) festgehaltenen Definition zeichnet. Dabei werden die verschiedenen Strings des Arrays hintereinandergezeichnet, so daß eine zusammenhängende Figur entsteht.

Nach dem Aufruf dieses Unterprogramms wird nun in Zeile 260 auf eine Taste gewartet. Ist dieser Tastendruck erfolgt, so wird das nunmehr alte, auf dem Bildschirm befindliche Shape durch nochmaligen Aufruf von 610 gelöscht (Invertiermodus), um sofort die Auswertung des Tastendrucks anzuschließen (ab Zeile 320).

Hier werden alle unter den beiden Modi erlaubten Tasten getestet (s.o.) und entsprechend ausgeführt. Dabei ist speziell die Zeile 550 interessant. In den fünf Zeilen davor wurden je nach Bewegungsrichtung einzelne Ziffern (oder der Punkt) an den aktuellen String S$(n) angehängt. Ist der String voll (255 Zeichen), so wird automatisch der nächste String des Arrays "eingeschaltet", der ja ab Zeile 610 an den vorherigen nahtlos angehängt wird.

Dieses Programm sollten Sie erweitern. Sie könnten z.B. den Graphikcursor sichtbar machen und ein Verbessern bzw. Editieren erlauben. So wird es Ihnen ein Einfaches sein, Ihre Shapes zu entwerfen.



Das folgende Programm dient zu Ihrer Belustigung und sollte Ihnen ein klein wenig die Möglichkeiten der Shapes vor Augen führen:

```
100 REM ******************
110 REM **              **
120 REM **  SHAPE-DEMO  **
130 REM **              **
140 REM ******************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 1,7
170 COLOR= 6,6
180 GOSUB 780              :REM BUCHSTABENDEFINITIONEN UEBERNEHMEN
190 REM
200 Y=70
210 FOR B=1 TO 10              :REM 10 BUCHSTABEN
220   R=A(B)                   :REM STARTROTATION AUS ARRAY
230   XZ=300 : ZM=4
240   FOR X=281 TO B*30-28 STEP -3
250     SCALE= R,3             :REM ROTATION
260     DRAW ZM,A$(B) ON XZ,Y  :REM BUCHSTABEN LOESCHEN
270     ZM=2                   :REM INVERTIERMODUS EIN
280     R=(R+4)/64
290     R=(R-INT(R))*64        :REM NEUER ROTATIONSWINKEL
300     SCALE= R,3             :REM SETZEN
310     DRAW ZM,A$(B) ON X,Y   :REM BUCHSTABEN ZEICHNEN
320     XZ=X                   :REM ALTEN X-WERT MERKEN
330   NEXT X
340 NEXT B
350 DRAW ,A$(11) ON 304,Y      :REM LETZTEN BUCHSTABEN NUR ZEICHNEN
360 REM
370 REM
380 TEXT ,"P R A E S E N T I E R T",70,100,0
390 REM
400 FOR T=1 TO 3000 : NEXT T   :REM ZEITSCHLEIFE
410 REM
420 REM
430 D=6 : SCALE= 0,4           :REM BREITE, GROESSE
440 REM
450 REM
460 REM ERSTES WORT:
470 REM
480 FOR X=0 TO 4         :REM ERSTEN 5 BUCHSTABEN "SUPER"
490   FOR Y=0 TO D       :REM BREITE D
500     DRAW ,K$(X) ON 46+44*X+2*Y,50+Y
510   NEXT Y
520 NEXT X
530 REM
540 REM
```

```
550 REM ZWEITES WORT:
560 REM
570 FOR X=0 TO 6       :REM NAECHSTE 7 BUCHSTABEN "GRAPHIK"
580   FOR Y=0 TO D
590     DRAW ,K$(X+5) ON 10+44*X+2*Y,110+Y
600   NEXT Y
610 NEXT X
620 REM
630 REM
640 REM DRITTES WORT:
650 REM
660 FOR X=0 TO 1       :REM LETZTE ZWEI BUCHSTABEN "64"
670   FOR Y=0 TO D
680     DRAW ,K$(X+12) ON 110+44*X+2*Y,170+Y
690   NEXT Y
700 NEXT X
710 REM
720 POKE 198,0 : WAIT 198,255
730 END
740 REM
750 REM
760 REM BUCHSTABENDEFINTIONEN:
770 REM
780 DIM K$(15),A$(11),A(11)
790 REM
800 K$( 0)="0.1.2.1.1.1.1.0.0.0.0.3.3.3.3.0.0.0.0.1.1.1.1.2.1"      :RE
M S
810 K$( 1)="00000000.2.2.2.2.2.2.2.2.1.1.1.1.0.0.0.0.0.0.0.0.0"     :RE
M U
820 K$( 2)=".0.0.0.0.0.0.0.0.1.1.1.1.2.2.2.2.3.3.3.3."              :RE
M P
830 K$( 3)="1111.3.3.3.3.0.0.0.0.1.1.1.333.0.0.0.0.1.1.1.1."        :RE
M E
840 K$( 4)=K$(2)+"1.2.1.2.1.2.1.2.1."                               :RE
M R
850 K$( 5)=".1.1.1.1.0.0.0.0.3.3.3.22223.0.0.0.0.0.0.0.0.1.1.1.1.2." :RE
M G
860 K$( 6)=K$(4)                                                    :RE
M R
870 K$( 7)=".0.0.0.0.0.0.0.0.1.1.1.1.2.2.2.2.2.2.2.2.0000.3.3.3.3."  :RE
M A
880 K$( 8)=K$(2)                                                    :RE
M P
890 K$( 9)=".0.0.0.0.0.0.0.0.2222.1.1.1.1.0000.2.2.2.2.2.2.2.2."     :RE
M H
900 K$(10)="1.1.1.1.1.33.0.0.0.0.0.0.0.0.33.1.1.1.1.1."             :RE
M I
910 K$(11)=".0.0.0.0.0.0.0.0.2222.0.1.0.1.0.1.0.1.22222222.3.0.3.0.3.0.3
.0."
920 K$(12)=".1.1.1.1.0.0.0.0.3.3.3.3.2.2.2.20000.0.0.0.1.0.1.1.1."   :RE
M 6
```

```
930 K$(13)="1111.0.0.0.0.0.22.1.1.3.3.3.3.3.0.0.1.0.1.0.1.0.1."      :RE
M 4
940 REM
950 REM
960 A$( 1)=".0.0.0.0.0.0.1.1.1.2.1.2.2.2.2.3.2.3.3.3"               :RE
M D
970 A( 1)=8
980 A$( 2)=".0.0.0.0.0.0.1.1.1.1.2.2.2.3.3.3.31111.2.2.2"           :RE
M A
990 A( 2)=48
1000 A$( 3)="111.0.0.0.0.0.0.3.3.3111.1.1.1"                        :RE
M T
1010 A( 3)=25
1020 A$( 4)=A$(2)                                                   :RE
M A
1030 A( 4)=0
1040 A$( 5)=""                                                      :RE
M SPACE
1050 A( 5)=0
1060 A$( 6)=".0.0.0.0.0.0.1.1.1.1.2.2.2.3.3.3.31111.2.2.2.3.3.3.3"   :RE
M B
1070 A( 6)=16
1080 A$( 7)=".1.1.1.1.133333.0.0.0.1.1.1.13333.0.0.0.1.1.1.1.1"     :RE
M E
1090 A( 7)=57
1100 A$( 8)=".1.1.1.13333.0.0.0.0.0.0.1.1.1.1"                      :RE
M C
1110 A( 8)=32
1120 A$( 9)=".0.0.0.1.2.1.2.1.2333000.1.0.1.0.1.0333.2.2.2"         :RE
M K
1130 A( 9)= 7
1140 A$(10)=A$(7)                                                   :RE
M E
1150 A(10)=48
1160 A$(11)=".0.0.0.1.2.1.2.1.2333000.0.0.0.1.1.1.2.2.2.3.3.3"      :RE
M R
1170 A(11)=60
1180 RETURN
```

In diesem Programm wurden verschiedene Buchstaben als Shapes definiert und diverse Effekte erzeugt. Wir wollen dieses Programm hier nicht näher erläutern. Schauen Sie es sich an, und Sie werden mit etwas Überlegung keine Probleme haben, die verschiedenen Vorgänge zu verstehen.

2.10 Beliebige Flächen ausfüllen - Das Juwel der Graphikbefehle

Setzen Sie sich erst einmal hin, bevor Sie weiterlesen, und halten Sie sich fest, das könnte Sie glatt vom Hocker hauen. Ich darf Sie mit einem Befehl bekannt machen, der Ihnen viel, sehr viel Arbeit abnimmt:

PAINT

Befehl:	PAINT zm,x,y
Beispiel:	PAINT ,160,100
Parameter:	zm: Zeichenmodus
	x: x-Koordinate des Testpunktes
	y: y-Koordinate des Testpunktes
Funktion:	Ausfüllen einer beliebigen umschlossenen Fläche

Sie geben ihm auf Ihrem Bildschirm eine beliebige Fläche vor - irgendetwas, was sie wollen. Dann tritt er in Aktion und Ihre Fläche beginnt zu strahlen.

Nun, bleiben wir auf dem Boden! Ihre Supergraphik bietet Ihnen etwas recht Hübsches: Angenommen, Sie wollen nicht nur mit dem inzwischen bekannten CIRCLE-Befehl einen einfachen Kreis (oder eine Ellipse) zeichnen, sondern diesen auch ausgefüllt wissen. Oder Sie haben sich eine schöne abgeschlossene Figur ausgedacht und haben das gleiche damit vor. In diesem Falle arbeiten Sie folgendermaßen: Sie zeichnen zunächst einmal die Umrandung der Ihnen vorschwebenden Figur auf den Bildschirm. Dabei achten Sie darauf, daß es keine Lücke in dieser Umrandung gibt. Nun bestimmen Sie einen einzigen Punkt, der innerhalb dieser Figur liegt und übergeben dessen Koordinaten in altbekannter Weise dem PAINT-Befehl als sogenannten *Testpunkt*. Das übrige erledigt sich von selbst: PAINT findet Ihre Umrandung und füllt die Figur mit Punkten vollständig auf. Der umgekehrte Fall gilt beim Löschen: hier erkennt PAINT alle gelöschten Punkte als Rand und löscht gleichfalls alle innerhalb liegenden Punkte (samt ehemaliger Umrandung).

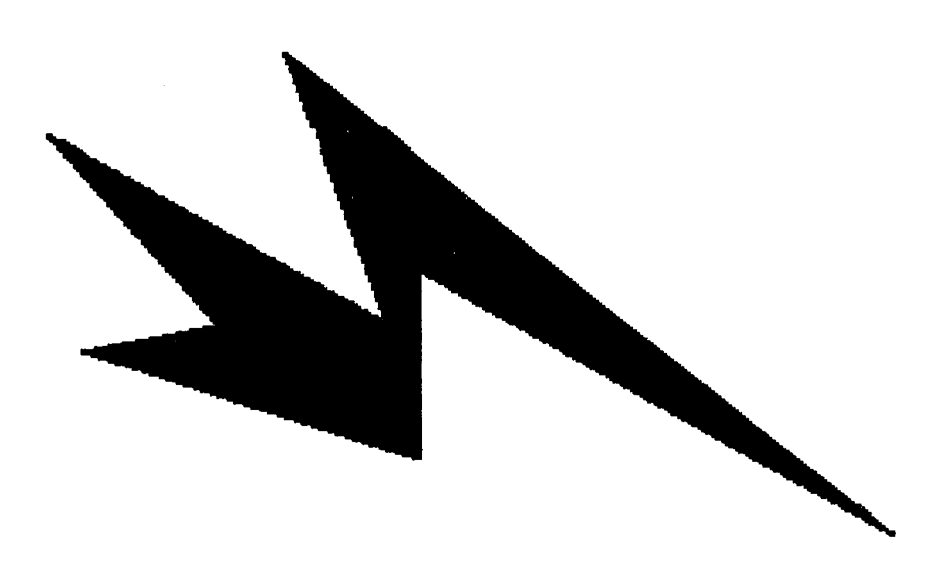
Natürlich können Sie auch einen Punkt außerhalb der Figur angeben. In diesem Falle wird der gesamte Bildschirm außer allen abgeschlossenen Figuren in ihm ausgefüllt (gelöscht).

Es kann schon einmal vorkommen, daß bestimmte Teile von Figuren (besonders in den spitzen Ecken) nicht ausgefüllt werden. Das jedoch liegt nicht am PAINT-Befehl, sondern vielmehr an der Art und Weise, wie Linien und Ellipsen etc. auf den Bildschirm gebracht werden. Wenn Sie genau hinschauen, werden Sie sehen, daß diese "Lücken" tatsächlich für sich bereits eigenständige und abgeschlossene kleine Flächen sind (manchmal nur einen Punkt groß).

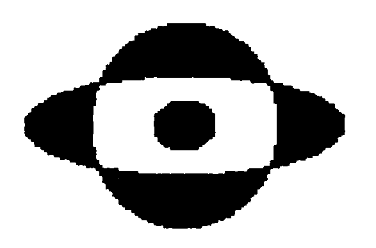
Die einzelnen Zeichenmodi sind zwar verfügbar, können allerdings oft nur mit Einschränkungen verwendet werden (z.B. der Punktier-Modus).

Falls die verschiedenen Figuren zu komplex sind, wird dies durch ein "OUT OF MEMORY ERROR" verkündet. Dies kann passieren, da sich der PAINT-Befehl alle Einbuchtungen usw. merken muß, um sie später noch auszufüllen. Wenn dieser als Stack organisierte Speicher voll ist, kann der Befehl nicht weiter ausgeführt werden.

Zur Demonstration einige Beispiele:







```
100 REM **********************
110 REM **                  **
120 REM **  PAINT-BEISPIEL  **
130 REM **                  **
140 REM **********************
150 REM
160 REM
170 REM ************
180 REM **        **
190 REM **  OVAL  **
200 REM **        **
210 REM ************
220 REM
230 GMODE 0,1 : GCLEAR
240 CIRCLE ,,160,100,50,80 :REM ELLIPSE ALS FIGURBEGRENZUNG
250 PAINT ,160,100         :REM AUSFUELLEN
260 FRAME ,1,10,10 TO 60,70:REM RECHTECK ALS FIGURBEGRENZUNG
270 PAINT ,11,11
280 REM
290 GOSUB 1240 :REM WARTEN
300 REM
310 REM
320 REM ***************
330 REM **           **
340 REM **  FIGUR 2  **
350 REM **           **
360 REM ***************
370 REM
380 GCLEAR
390 REM
400 REM ***************
410 REM FELD UMREISSEN:
420 REM ***************
430 REM
440 PLOT ,100,120 TO  50, 50 TO 150,120 TO 120,20
450 PLOT ,TO 300,199 TO 160,100 TO 160,170
460 PLOT ,TO 60,130 TO 100,120
470 PAINT  ,101,101      :REM FELD AUSWAEHLEN
480 GOSUB 1240
490 REM  PAINT 1,101,101 :REM FELD LOESCHEN
500 GOSUB 1240
510 REM
520 REM
530 REM ***************
540 REM **           **
550 REM **  FIGUR 3  **
560 REM **           **
570 REM ***************
580 REM
590 GCLEAR
600 REM
```

```
610 CIRCLE ,,160,100,30,40 :REM INNEN
620 CIRCLE ,,160,100,50,60 :REM MITTE
630 CIRCLE ,,160,100,70,80 :REM AUSSEN
640 PAINT ,160+29,100+39   :REM MITTLEREN STREIFEN AUSFUELLEN
650 PAINT ,0,0             :REM AUSSERES FELD AUSFUELLEN
660 REM
670 GOSUB 1240
680 REM
690 REM
700 REM *****************
710 REM **             **
720 REM **  BLUEMCHEN  **
730 REM **             **
740 REM *****************
750 REM
760 GCLEAR
770 CIRCLE ,,160,100,10,10
780 CIRCLE ,,160,100,30,40
790 CIRCLE ,,160,100,50,20
800 PAINT ,   160,100
810 PAINT ,   160,100+38
820 PAINT ,   160,100-38
830 PAINT ,160+48,100
840 PAINT ,160-48,100
850 GOSUB 1240
860 REM
870 REM
880 REM ***************
890 REM **           **
900 REM **  FIGUR 5  **
910 REM **           **
920 REM ***************
930 REM
940 GCLEAR
950 CIRCLE ,,160,100,70,90
960 CIRCLE ,,125, 40,20,25
970 CIRCLE ,,195, 40,20,25
980 CIRCLE ,,160, 93,15,50
990 CIRCLE ,,160,100,62,62,147,213
1000 CIRCLE ,,160,100,50,75,135,227
1010 CIRCLE ,,130, 48,10,10
1020 CIRCLE ,,190, 48,10,10
1030 CIRCLE ,,132, 52, 4, 4
1040 CIRCLE ,,188, 52, 4, 4
1050 CIRCLE ,,230, 85,20,60,305,205
1060 CIRCLE ,, 90, 85,20,60,155, 55
1070 CIRCLE ,,230, 85,10,30,305,205
1080 CIRCLE ,, 90, 85,10,30,155, 55
1090 PLOT ,180,185 TO 180,199
1100 PLOT ,140,185 TO 140,199
1110 FRAME ,1,145, 37 TO 175,43
```

```
1120 PAINT ,160+20,100
1130 PAINT ,160,30
1140 PAINT ,130,48
1150 PAINT ,190,48
1160 PAINT ,230+15,85
1170 PAINT , 90-15,85
1180 WAIT 198,255 : END
1190 REM
1200 REM
1210 REM WARTESCHLEIFE:
1220 REM ***************
1230 REM
1240 FOR T=1 TO 600 : NEXT T
1250 RETURN
```

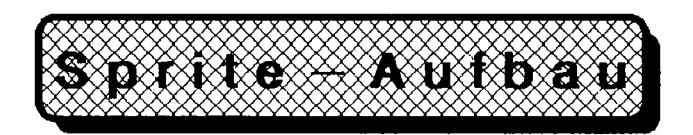
2.11 Die Welt der Sprites

Eines der hervorstechendsten Merkmale Ihres Commodore 64 ist die Fähigkeit, Sprites darzustellen. Sprites sind eigenständige kleine Graphiken, die unabhängig voneinander und von dem übrigen Bildschirminhalt in dem Text- oder Graphikfenster bewegt werden können. Im Unterschied zu Shapes werden Sprites hardwaremäßig realisiert und sind dadurch extrem schnell zu handhabende Objekte. Insgesamt haben Sie die Möglichkeit, 8 Sprites gleichzeitig auf den Bildschirm zu bringen.

Sprites können bezüglich Ihrer Farbe, Ihrer Größe und der Priorität vor den Hintergrundzeichen und auch gegeneinander variiert werden. Kollisionen zwischen Sprites untereinander und mit dem Hintergrund lassen sich feststellen.

All diese Funktionen können sehr leicht mit Hilfe des VIC (Videocontroller 6567) und seinen Registern realisiert werden. Supergraphik erleichtert zudem die Handhabung dieser Sprites noch enorm, so daß Sie sämtliche Funktionen leicht und überschaubar auf wenige Befehle komprimiert vor sich liegen sehen.

Zunächst einmal wollen wir uns mit dem Aufbau der Sprites befassen: Jedes Sprite besteht aus 504 Punkten, die Sie einzeln setzen können. Verwendet wird dabei eine 24x21-Punktematrix, d.h. ein Sprite ist 24 Punkte breit und 21 Punkte hoch. Innerhalb dieses Bereiches können Sie nun die unterschiedlichsten Graphiken oder Figuren erstellen. Wir können diesen Sachverhalt in einer kleinen Skizze darlegen:



	Spalte 0								Spalte 1								Spalte 2							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0																								
Byte 3																								
Byte 6																								
Byte 9																								
Byte 12																								
Byte 15																								
Byte 18																								
Byte 21																								
Byte 24																								
Byte 27																								
Byte 30																								
Byte 33																								
Byte 36																								
Byte 39																								
Byte 42																								
Byte 45																								
Byte 48																								
Byte 51																								
Byte 54																								
Byte 57																								
Byte 60																								

Dabei wird jede Zeile in drei Spalten zu je 8 Punkten unterteilt. Diese jeweils 8 Punkte stellen im Speicher 8 Bit und damit ein Byte dar (Werte von 0-255). D.h. die ersten 8 Punkte der ersten Zeile werden zu einem Byte zusammengefaßt. Dabei stellt jeder gesetzte Punkt eine 1 und jeder nicht gesetzte Punkt eine 0 dar. Die 8 hintereinanderstehenden Einsen und Nullen werden dann als Dual- bzw. Binärzahl verstanden (s. Anhang) und in die dezimale Schreibweise übersetzt.

Auf die gleiche Weise verfährt man mit den zweiten und dritten 8 Punkten einer Zeile. Danach widmet man sich der nächsten Zeile und so fort. Auf diese Weise erhalten wir insgesamt 3x21=63 Werte, die die Definition eines Sprites darstellen. Ganz genau wird das in dem Kapitel Hardwaregrundlagen erläutert.

Sie haben die Möglichkeit, die Farbe der gesetzten Punkte getrennt von dieser Definition aus 16 möglichen Farben zu bestimmen (s.u.).

Nicht jedes Sprite jedoch besitzt diesen 24x21-Punkte-Aufbau. Sie können jeweils zwischen hochauflösenden und sogenannten Multicolor-Sprites wählen. Erstere besitzen den gerade geschilderten Aufbau. Die Multicolor-Sprites hingegen werden ähnlich der Multicolor-Graphik gebildet. Aus diesem Grunde besitzt ein solches Sprite in x-Richtung die halbe Auflösung. Hier befinden sich also nur 12, jedoch doppelt breite Punkte in einer Zeile. Dafür kann ein Gebilde aus insgesamt vier verschiedenen Farben (mit der Hintergrundfarbe) zusammengesetzt sein, während, wie gesagt, ein hochauflösendes (HGR-) Sprite nur zwei Farben beinhaltet.

Im Speicher sieht die Definition der Multicolor-Sprite haargenauso aus, wie die eines HGR-Sprites, mit dem einen Unterschied: Beim MC-Sprite stellen jeweils zwei Bits (also zwei Einsen oder Nullen) einen doppelt breiten Punkt dar. Da zwei Bits aber vier verschiedene Werte annehmen (00,01,10,11) können, stellen diese vier Werte die vier möglichen Farben des Punktes dar. Diese vier Farben sind in verschiedenen Registern des VIC untergebracht (s. Hardwarkapitel). Einschränkend muß jedoch gesagt werden, daß lediglich die Farbe 2 für alle Sprites unterschiedlich sein kann. Die anderen Farben (Farben 1 und 3 neben der Hintergrundfarbe) sind jeweils für alle Sprites gleich, da sie aus identischen Registern gewonnen werden. Entgegen den Angaben des CBM 64 Benutzerhandbuches können auch in Multicolor sämtliche 16 Farben zur Erstellung Ihrer Figuren verwendet werden.

Neben der Farbe und der Definition der Sprites können noch einige Dinge verändert werden: Sie haben die Möglichkeit, Ihre Gebilde in x-, in y- oder in beide Richtungen um den Faktor 2 zu vergrößern. D.h. Ihr Sprite wird doppelt breit, doppelt hoch oder beides.

Zudem geben Sie an, welche Priorität das jeweilige Sprite vor dem Hintergrund besitzt, d.h. ob Ihr Sprite nun vor den Zeichen oder der übrigen Graphik steht (oder fliegt), sie also verdeckt, oder ob es sich dahinter befindet, durch diese also verdeckt wird. Gleichzeitig bestimmen Sie durch die Nummer eines Sprites (jedes der acht möglichen Sprites besitzt eine Nummer von 0 bis 7) die Priorität der Sprites untereinander. Diejenigen Sprites mit der höheren Nummer besitzen die höhere Priorität, verdecken bei Überschneidungen also stets die Sprites mit einer niedrigen Nummer. So können Sie recht einfach 3-dimensionale Effekt erzeugen.

Doch damit nicht genug. Ihr Commodore 64 bietet Ihnen noch eine Reihe weiterer Bonbons, die Ihnen das Leben versüßen und auf die Sie ganz schön scharf sein werden, wenn Sie wissen, um was es geht! Ihr Computer registriert nämlich jegliche Berührung (oder Kollision) eines Sprites mit einem Hintergrundzeichen (oder überhaupt mit einem Punkt auf dem Bildschirm) oder einem anderen Sprite. Dieses Ereignis wird ebenfalls in verschiedenen Registern des VIC abgelegt, was Sie jedoch nicht weiter zu interessieren braucht, da Ihnen die Supergraphik diese Arbeit weitgehend abnimmt. Hierbei sind nur einige Dinge zu beachten, die unter den jeweiligen Befehlen (IF#...THEN) aufgeführt sind. Wichtig ist lediglich die Tatsache, daß zwischen Kollisionen Hintergrund-Sprite und Sprite-Sprite differenziert werden kann. Im letzteren Fall kann noch entschieden werden, welche der acht Sprites zusammengestoßen sind.

Sie sehen also, mit den Sprites werden Ihnen Möglichkeiten in die Hand gegeben, die Sie erst bei der Erprobung und Anwendung völlig ausschöpfen können. Sehr geeignet sind diese Figuren, die ja hardwaremäßig erzeugt werden und deswegen sehr schnell zu handhaben sind, zur Produktion von schönen Spielen oder auch Laufschriften. Lassen Sie sich etwas einfallen. Die einzelnen Beispielprogramme bei den Erläuterungen der einzelnen Befehle sollen Ihnen hier Ansätze und Ideen vermitteln, die bei Ihrem nächsten Programm vielleicht Anwendung finden.

2.11.1 Unser erstes Sprite

Bevor wir tatsächlich unser erstes Sprite auf den Bildschirm zaubern können, müssen wir leider eine ganze Menge Dinge festlegen. Das beginnt bei der Definition des Sprites und hört bei Farben und Größe noch nicht auf. Der erste Befehl, den wir hierzu vorstellen, hat dann auch noch eigentlich gar nichts mit Sprites direkt zu tun:

SREAD

Befehl:	SREAD str$
Beispiel:	SREAD A$
Parameter:	str$:Zielstringvariable
Funktion:	Einlesen von 63 Spritedaten

Dieser Befehl ist eine sehr nützliche Hilfe bei der Definition von Sprites. Sprites werden mit Hilfe von Strings, die die erforderlichen 63 Bytes als ASCII-Werte enthalten, definiert (s. SDEFINE). Statt nun umständlich aus DATA-Zeilen Byte für Byte einzulesen, geben Sie einfach z.B. SREAD A$ ein und schon sind sie im Speicher A$. Dieser Befehl ersetzt also die BASIC-Zeile:

```
FOR X=1 TO 63 : READ B : A$=A$+CHR$(B) : NEXT X
```

Man spart also Platz, Aufwand und Zeit (die Daten werden etwa drei mal so schnell eingelesen). Die Fehlermeldungen entsprechen denen eines normalen READ-Befehls. Die Anwendung von SREAD wird in weiter unten stehenden Programmen demonstriert. Dieses Kommando kann selbstverständlich auch für andere Zwecke verwendet werden.

Zusätzlich zu den Befehlen der Supergraphik befindet sich auf Ihrer Diskette ein kompletter Spritegenerator, der Ihnen hilft, sehr komfortabel Ihre individuellen Sprites zu generieren und im Kapitel "Supergraphik intern" beschrieben ist.

Im folgenden bedeutet "s" immer die Spritenummer und geht von 0-15(!), x und y sind nun die Spritekoordinaten:

```
x: 0-511
y: 0-255
```

Wie Sie sehen, stimmen hier die Koordinaten nicht mit denen der hochauflösenden Graphik oder der Graphik allgemein überein. Sie können Sprites nämlich auch außerhalb des Bildschirms positionieren, um ein Sprite langsam hinter dem Bildschirmrand verschwinden lassen zu können. Vielleicht erstaunt Sie auch der Wertebereich der Spritenummern von 0-15, wo wir doch nur 8 Sprites gleichzeitig auf dem Bildschirm darstellen können. Mit Supergraphik jedoch ist es Ihnen möglich, auch 16 völlig unterschiedliche Sprites gleichzeitig auf dem Bildschirm darzustellen. Dies wird durch die Interrupttechnik der Supergraphik möglich. Mehr davon aber beim Befehl SPOWER.

SDEFINE

Befehl:	SDEFINE s,str$
Beispiel:	SDEFINE 9,A$
Parameter:	s: Spritenummer (0-15)
	str$: Definitionsstring
Funktion:	Spritedefinition - Einschalten/Ausschalten

Dies ist unser erster richtiger Sprite-Befehl: Mit diesem Befehl definieren Sie ein Sprite, dessen Daten in einem String str$ wie folgt untergebracht sind: Im Vorspann dieses Abschnitts haben wir kurz den Aufbau eines Sprites im Speicher kennengelernt. Dabei haben wir festgestellt, daß sich eine Spritedefinition aus 8x21=63 Bytes, also auch 63 Werten zusammensetzt, die jeweils das Bitmuster des Objektes darstellen (s. auch Hardwarekapitel). Diese 63 Bytes werden nun als ASCII-Werte von 0-255 hintereinander in einen String gepackt, der hier als Definitionsstring verwendet wird.

Diese Daten werden dem Sprite mit der Nummer s zugeordnet. Grundsätzlich können alle 16 möglichen Sprites ein völlig anderes Aussehen haben. Die Sprites mit den Nummern 8-15 jedoch werden erst eingeschaltet, wenn Sie den Befehl SPOWER eingegeben haben. Doch damit muß ein Sprite noch nicht unbedingt sichtbar sein! Zunächst müssen weitere Parameter und die Lage auf dem Bildschirm bestimmt werden.

Allgemein kann aber gesagt werden, daß Sprites nur in der HGR- bzw. MC-Anzeige verfügbar sind (dies hängt mit der Verschiebung des Video-RAM zusammen).

Wollen Sie ein Sprite wieder ausschalten, so darf der String kein Zeichen enthalten (z.B.: SDEFINE 0,"" - Sprite 0 wird ausgeschaltet).

Das folgende kleine Demonstrationsprogramm soll Ihnen die Anwendung der Befehle SREAD und SDEFINE veranschaulichen. Damit überhaupt etwas auf dem Bildschirm erscheint, mußten wir in den beiden Zeilen 180 und 200 etwas vorgreifen und noch unbekannte Befehle verwenden. Dies sollte Sie aber nicht stören.

```
100 REM ***************
110 REM **           **
120 REM **  SPRITES  **
130 REM **           **
140 REM ***************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 SREAD A$        :REM SPRITEDEFINITION AUS DATAS NACH A$ UEBERNEHMEN
180 SMODE 0,3,1     :REM SPRITE 0: MODUS 3, FARBE GELB
190 SDEFINE 0,A$    :REM SPRITE 0 EINSCHALTEN MIT DEF. IN A$
200 SSET 0,100,150 :REM SPRITE POSITIONIEREN
210 WAIT 198,255
1000 REM
1010 REM
1020 REM SPRITEDEFINITION:
1030 REM
1040 DATA   0,  7,  0
1050 DATA  56, 13,128
1060 DATA  25, 31,224
1070 DATA 126, 31,204
1080 DATA  25, 12,252
1090 DATA  56,  7,  0
1100 DATA   0, 14,  0
1110 DATA   0, 14,  0
1120 DATA   0, 15,128
1130 DATA   0, 14,192
1140 DATA   0, 30, 96
1150 DATA   0, 62, 48
1160 DATA   0, 46,  0
1170 DATA   0, 79,  0
1180 DATA   0,155,128
1190 DATA   1, 25,192
1200 DATA   2, 24, 96
1210 DATA   3, 24, 96
1220 DATA 127, 24,120
1230 DATA 188, 24,  0
1240 DATA  36, 30,  0
```

Hier wird in Zeile 170 die gesamte Definition des Sprites in den DATA-Zeilen als ASCII-Werte in den String A$ übertragen. Die Definition in A$ dient dann in Zeile 190 als Grundlage des Befehls SDEFINE zum Definieren und Einschalten des Sprites mit der Nummer 0.

Die DATA-Zeilen sind jeweils so angeordnet, daß eine DATA-Zeile einer Zeile des Sprites entspricht. Nehmen wir uns die ersten beiden DATA-Zeilen heraus (Zeilen 1040/1050), so können wir aus ihnen ablesen, welche Punkte in der ersten beiden Zeilen des Sprites gesetzt und welche gelöscht sind:

```
Erste Zeile: DATA-Zeile:       0,       7,       0
             Bit-Muster: 00000000 00000111 00000000
             Sprite    :               ...


Zweite Zeile: DATA-Zeile:      56,      13,      128
              Bit-Muster: 00111000,00001101,10000000
              Sprite    :   ...        .. . .
```

Das Bit-Muster ist lediglich die Übersetzung des Wertes in der DATA-Zeile ins Duale (Binäre) Zahlensystem (s. Anhang). Aus den Nullen und Einsen ergeben sich dann die gesetzten Punkte im Sprite.

Kommen wir zu einem weiteren Sprite-Befehl, der eine ganze Reihe von Möglichkeiten des Rechners bei der Spritedarstellung ausschöpft:

SMODE

Befehl:	SMODE s,m,f1 (,f2,f3)	
Beispiel:	SMODE 1,3,7	
oder	SMODE 1,8,7,2,14	
Parameter:	s:	Spritenummer
	m:	Spritemodus (s.u.)
	f1:	Farbe 1 (individuell)(0-15)
	f2:	Farbe 2 (Multicolor-einheitlich)(0-15)
	f3:	FArbe 3 (Multicolor-einheitlich)(0-15)
Funktion:	Bestimmen der Spriteeigenschaften	

Dieser Befehl definiert das weitere Aussehen des Sprites mit der Nummer s. m gibt dabei den Spritemodus an, wobei gilt: Werte für m gehen von 0 bis 15:

m MC	m HGR	Vergrößerung x-Richtung	Vergrößerung y-Richtung	Priorität vor dem Hintegrund
8	0	nein	nein	ja
9	1	ja	nein	ja
10	2	nein	ja	ja
11	3	ja	ja	ja
12	4	nein	nein	nein
13	5	ja	nein	nein
14	6	nein	ja	nein
151	7	ja	ja	nein

Wie Sie sehen, haben Sie die Möglichkeit, 16 verschiedene Einstellungen vorzunehmen. Dabei bestimmen die Werte 8-15 ein Multicolor-Sprite und fordern damit noch die Angabe der Werte f2 und f3. Diese Werte geben die Farben 2 und 3 der Multicolor-Sprites an, die allerdings für alle Sprites gleich sind (die Spritegruppen 0-7 und 8-15 jedoch können auch in dieser Eigenschaft differieren). Bei hochauflösenden Sprites genügt die Angabe von f1. f1 definiert (auch in MC) die Farbe 1, die für alle Sprites unterschiedlich sein kann. Für die Farbnummern gelten folgende Zuordnungen:

Farbtabelle:

f1,f2,f3 (Farbnr.)	Farbe	f1,f2,f3 (Farbnr.)	Farbe
0	schwarz	8	orange
1	weiß	9	braun
2	rot	10	hellrot
3	zyanblau	11	dunkelgrau
4	violett	12	mittelgrau
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	hellgrau

Selbst mit diesem Befehl braucht unser Sprite noch nicht sichtbar zu sein, denn es kann ja irgendwo auf dem oder außerhalb des Bildschirms positioniert sein. Aus diesem Grunde wollen wir uns noch schnell das Wissen über einen weiteren Befehl aneignen, der diesen Mangel behebt.

SSET

	Befehl:	SSET s,x,y (,sp)
Beispiel:	SSET 1,100,120	
oder	SSET 1,100,120,4	
Parameter:	s:	Spritenummer
	x:	Sprite-x-Koordinate
	y:	Sprite-y-Koordinate
	sp:	Geschwindigkeit für Spritebewegung
Funktion:	Positionieren eines Sprites	

Durch diesen Befehl sind Sie in der Lage, ein Sprite s (0-15) an die Position x,y auf dem Graphikbildschirm zu beordern, wobei die Koordinate stets den Ort der rechten oberen Ecke des Sprites bezeichnet. Sie können mit Ihrem Sprite bekanntlich aus dem Bildschirmfenster herausfahren. Die Sprite-Koordinate stimmt also nicht mit der Graphikkoordinate überein - sie ist relativ zu ihr um einige 10 Punkte verschoben. Finden Sie doch einmal heraus, um wieviele; so lernen Sie die Spritebefehle näher kennen. Nebenbei: die Auflösung finden Sie auch im Hardwarekapitel.
Mit sp (speed) können Sie die Geschwindigkeit dieses Sprites für einen späteren SSET...TO... - Befehl vorprogrammieren, doch darüber unterhalten wir uns noch etwas weiter unten.

Im folgenden Programm können Sie sich einmal anschauen, wie Sie mit den beiden gerade besprochenen Befehlen umgehen können und welche Wirkung die einzelnen Parameter besitzen:

```
100 REM ******************
110 REM **              **
120 REM **  SMODE,SSET  **
130 REM **              **
140 REM ******************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0 : SCOL= 1,9
170 SREAD A$     :REM SPRITEDEFINITION AUS DATAS NACH A$ UEBERNEHMEN
180 SDEFINE 0,A$ :REM SPRITE 0 EINSCHALTEN MIT DEF. IN A$
```

```
190 X=100 : Y=150     :REM SSET-PARAMETER
200 MODUS=3 : FARBE=1 :REM SMODE-PARAMETER INIT
210 FILL B 15,,50,50 TO 150,140
220 FL=0              :REM FLAG
230 REM
240 REM
250 REM HAUPTSCHLEIFE:
260 REM
270 REM
280 REM WERTE AUSGEBEN:
290 FILL 1,10,183 TO 170,190
300 TEXT ,"FARBE:"+STR$(FARBE)+" MODUS:"+STR$(MODUS),10,190,1
310 REM
320 SMODE 0,MODUS,FARBE         :REM WERTE REALISIEREN
330 SSET 0,X,Y
340 GET T$ : IF T$="" THEN 340
350 T=ASC(T$)
360 IF T$="Q" THEN END
370 IF T$="W" THEN FL=ABS(FL-1) :REM FLAG WECHSELN
380 IF FL=  1 THEN 580
390 REM
400 REM
410 REM BEI FLAG=0:
420 REM
430 IF T= 29 THEN FARBE=FARBE+1 :REM RECHTS
440 IF T=157 THEN FARBE=FARBE-1 :REM LINKS
450 IF T= 17 THEN MODUS=MODUS+1 :REM HOCH
460 IF T=145 THEN MODUS=MODUS-1 :REM RUNTER
470 REM
480 REM WERTE UEBERPRUEFEN:
490 IF FARBE>15 THEN FARBE= 0
500 IF FARBE< 0 THEN FARBE=15
510 IF MODUS>15 THEN MODUS= 0
520 IF MODUS< 0 THEN MODUS=15
530 GOTO 290
540 REM
550 REM
560 REM BEI FLAG=1:
570 REM
580 IF T= 29 THEN X=X+10        :REM RECHTS
590 IF T=157 THEN X=X-10        :REM LINKS
600 IF T= 17 THEN Y=Y+10        :REM HOCH
610 IF T=145 THEN Y=Y-10        :REM RUNTER
620 REM
630 REM WERTE UEBERPRUEFEN:
640 IF X>511 THEN X=  0
650 IF X<  0 THEN X=511
660 IF Y>255 THEN Y=  0
670 IF Y<  0 THEN Y=255
680 GOTO 330
690 REM
```

```
700 REM
710 REM SPRITEDEFINITION:
720 REM
730 DATA   0, 24,  0
740 DATA   0, 60,  0
750 DATA   0, 60,  0
760 DATA   0, 24,  0
770 DATA   6, 24, 96
780 DATA   3, 24,192
790 DATA   1,153,128
800 DATA   0,219,  0
810 DATA  48,126,  6
820 DATA 127,255,255
830 DATA 127,255,255
840 DATA  48,126,  6
850 DATA   0,219,  0
860 DATA   1,153,128
870 DATA   3, 24,192
880 DATA   6, 24, 96
890 DATA   0, 24,  0
900 DATA   0, 60,  0
910 DATA   0, 60,  0
920 DATA   0, 24,  0
930 DATA   0,  0,  0
```

Dies ist wieder eines der Programme, bei denen Sie direkt auf dem Bildschirm beobachten können, was die einzelnen Parameter der Befehle SMODE und SSET bewirken. Lassen Sie uns kurz die Möglichkeiten darlegen, die Sie besitzen, wenn Sie das Programm gestartet haben. Nach dem Start wird ein Sprite, das in den Zeilen 730-930 in DATA-Zeilen festgehalten wurde auf den Bildschirm gebracht. Das Sprite erscheint über einem gestreiften Feld, durch das die Priorität des Sprites vor dem Hintergrund deutlich wird. Etwas weiter unten werden die aktuellen Einstellungen von Sprite-Modus und Farbe angezeigt, die beide durch den Befehl SMODE festgelegt werden können. Das Programm reagiert nun auf folgende Tastenbetätigungen:

Q - Programmende
W - Programmmoduswechsel

Bei Modus 0:

CRSR rechts	- Farbnummer des Sprites erhöhen
CRSR links	- Farbnummer des Sprites erniedrigen
CRSR hoch	- Spritemoduszahl erhöhen
CRSR runter	- Spritemoduszahl erniedrigen

Bei Modus 1:

CRSR rechts	- Sprite nach rechts bewegen
CRSR links	- Sprite nach links bewegen
CRSR hoch	- Sprite nach oben bewegen
CRSR runter	- Sprite nach unten bewegen

Sie können also das Sprite über den gesamten Bildschirm bewegen und sowohl Farbe als auch Spritemodus (Größe, Priorität, MC/HGR) verändern. Dabei werden Ihnen ständig die aktuellen SMODE-Parameter angezeigt. Diese Anzeige erfolgt in Zeile 300, nachdem das Anzeigefeld durch einen FILL-Befehl gelöscht wurde (Zeile 290). Zeile 210 mag Ihnen vielleicht etwas seltsam vorkommen. Durch sie wird jedoch das gestreifte Feld unter (oder über - je nach Priorität) dem Sprite gezeichnet. Hierzu wird die B-Erweiterung der Zeichenbefehle verwendet. Der Rest des Programmes sollte Ihnen nun kein großes Kopfzerbrechen mehr machen, da die Struktur bereits aus anderen Programmen bekannt ist.

Bevor wir aus dem Probierstadium heraustreten, sollten wir noch eben einen weiteren Befehl kennenlernen, der uns ungeahnte Möglichkeiten eröffnet.

2.11.2 Der Trick beim Zeichentrick

SSET...TO...

Befehl:	SSET s,x1,y1 TO x2,y2 (,sp)	
Beispiel:	SSET 1,100,120 TO 50,40	
oder	SSET 1,100,120 TO 50,40,4	
Parameter:	s:	Spritenummer
	x1:	Sprite-x-Startkoordinate
	y1:	Sprite-y-Startkoordinate
	x2:	Sprite-x-Zielkoordinate
	y2:	Sprite-y-Zielkoordinate
	sp:	Geschwindigkeit für Spritebewegung
Funktion:	Bewegen eines Sprites von x1,y1 nach x2,y2	

So, nun geht es los!

Mit diesem Befehl - so unscheinbar er aussieht - können Sie Welten bzw. Sprites bewegen. Er läßt jedes beliebige Sprite von den angegebenen Ausgangskoordinaten mit der Geschwindigkeit sp in Form einer Geraden - wenn Sie wollen - quer über den Bildschirm zu den Endkoordinaten (hinter TO angegeben) wandern.

Kurz nachdem Sie den Befehl gegeben haben, wird schon der nächste abgearbeitet, während das Sprite seelenruhig über den Bildschirm tuckert oder rast - je nach Geschwindigkeit. Sie können auch dann ein zweites, drittes, viertes etc. Sprite völlig unabhängig von den anderen und gleichzeitig kreuz und quer verschicken.

Wie ist das nun möglich? Der Befehl übergibt die notwendigen Informationen der stets im Abstand von 1/60 Sekunde durchlaufenen IRQ-Routine (s.o. unter "Graphikfenster durch Rasterinterrupt oder im Hardwarekapitel). Diese versetzt Ihr Sprite nun pro Druchlauf um die mit sp angegebene Zahl von Punkten in Richtung Zielkoordinate, d.h. Ihr Sprite bewegt sich also mit einer Geschwindigkeit von 20*sp Punkten pro Sekunde über den Bildschirm.

Anmerkung: Wieviele Punkte dabei durchlaufen werden, können Sie einfach erfahren, indem Sie die Punkte einer entsprechenden Geraden (Linie) von der Ausgangsposition zum Ziel zählen (mit dem T-Sekundärbefehl (s.o.)).

Lassen Sie sp weg, so wird die dem jeweiligen Sprite vorher zugeordnete Geschwindigkeit übernommen (sp=0 entspricht sp=256).

Bewegen Sie sehr viele Sprites gleichzeitig, kann es dazu kommen, daß ihr Programm durch die vielen langen IRQ-Unterbrechungen zur Spritebewegung langsamer wird, aber es läuft weiter!

Verwenden Sie bitte das unter SDEFINE angegebene Programm und geben Sie folgende Zeilen ein:

```
200 SSET 0,0,0 TO 370,250,3
205 FRAME ,100,0,0 TO 319,199
```

Ihr Sprite bewegt sich also nun von links oben nach rechts unten in die Ecke und verschwindet dort. Gleichzeitig(!) wird ein dicker Rahmen gezeichnet.

Sehr wirkungsvoll ist nun auch der IF#C THEN ...-Befehl (s.u.). Eine Anmerkung: während des CIRCLE-Befehls steht aus Zeitgründen jedes Sprite und es kann auch keine Tastatureingabe gemacht werden.

SSET TO...

Befehl:	SSET s TO x2,y2 (,sp)	
Beispiel:	SSET 1 TO 50,40	
oder	SSET 1 TO 50,40,4	
Parameter:	s:	Spritenummer
	x2:	Sprite-x-Zielkoordinate
	y2:	Sprite-y-Zielkoordinate
	sp:	Geschwindigkeit für Spritebewegung
Funktion:	Bewegen eines Sprites von der momentanen Position nach x2,y2	

Dieses Kommando führt die gleiche Funktion aus, wie der vorherige Befehl. Dabei wird das angesprochene Sprite von der derzeitigen Position aus bewegt. Sie erkennen die Ähnlichkeit des SSET-Befehls mit dem PLOT-Befehl. Auf diese Gemeinsamkeit wurde bewußt geachtet. Erinnern Sie sich noch an den reinen SSET-Befehl? Dort konnten Sie bereits die Geschwindigkeit sp vorgeben. Sie könnten also mit einem einfachen SSET-Befehl alles Vorbereiten (Position und Geschwindigkeit), um das Sprite mit dem Befehl SSET TO..: weiter zu

bewegen. Mit diesem Befehl brauchen Sie sich also nicht zu merken, an welcher Stelle welches Ihrer Sprites steht, wenn Sie es weiterbewegen wollen.

SWAIT

Befehl:	SWAIT s
Beispiel:	SWAIT 2
Parameter:	s: Spritenummer
Funktion:	Warten bis Sprite s steht

Dieser Befehl wurde eingerichtet, damit Sie eine Kontrolle in Ihre Spritebewegung bekommen. Da das Programm normalerweise auch dann weiterläuft, wenn Sie ein Sprite bewegen, wartet Ihr Programm bei diesem Befehl so lange, bis das Sprite mit mit der Nummer s steht, also nicht mehr bewegt wird. Steht es bereits still, hat der Befehl keine besondere Wirkung.

Fügen Sie zu dem unter SSET genannten Programm folgende Zeile hinzu:

```
206 SWAIT 0 : CIRCLE ,,159,99,20,30
```

Erst nachdem Sprite 0 nicht mehr in Bewegung ist, wird die Ellipse gezeichnet.

Das folgende Programm soll Ihnen die Anwendung der Spritebefehle und ihr Zusammenwirken demonstrieren. Geben Sie es ruhig einmal ein und lassen Sie sich überraschen.

```
100 REM ******************
110 REM **              **
120 REM **  SPRITEDEMO  **
130 REM **              **
140 REM ******************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 1,0 : COLOR= 6,6
170 SREAD A$ :REM SPRITEDEFINITION AUS DATAS NACH A$ UEBERNEHMEN
180 REM
190 REM
200 REM 8 SPRITES AKTIVIEREN:
```

```
210 REM
220 FOR S=0 TO 7
230   SMODE S,0,1.17*(7-S)  :REM MODUS 0, FARBEN UNTERSCHIEDLICH
240   SSET S,(S-4*INT(S/4))*91,INT(S/4)*250,1 :REM POSITIONIEREN
250   SDEFINE S,A$          :REM EINSCHALTEN
260 NEXT S
270 REM
280 REM 8 SPRITES IN BEWEGUNG SETZEN:
290 REM
300 FOR S=0 TO 7
310   SSET S TO 160,130
320 NEXT S
330 REM
340 FOR X=1 TO 215
350   SCALE= X,X/2
360   DRAW ,".0" ON X,40*SIN(X/30)+100
370 NEXT X
380 REM
390 SWAIT 0 :REM AUF SPRITE 0 WARTEN
400 TEXT ,"SPRITES FOR EVER!",100,180,0
410 REM
420 FOR X=1 TO 1000 : NEXT X
430 REM
440 REM
450 REM ALLE SPRITES AUSSCHALTEN:
460 REM
470 FOR S=0 TO 7
480   SDEFINE S,""
490 NEXT S
500 REM
510 WAIT 198,255
520 REM
530 REM
540 REM SPRITEDEFINITION:
550 REM
560 DATA   0,  0,  0
570 DATA   0,  0,  0
580 DATA   0,  0,  0
590 DATA   2,  0, 64
600 DATA   1,  0,128
610 DATA   0,129,  0
620 DATA   0, 66,  0
630 DATA   0, 60,  0
640 DATA   0,126,  0
650 DATA   0,195,  0
660 DATA   1,141,128
670 DATA   3, 44,192
680 DATA  31,255,248
690 DATA  62,153,124
700 DATA 125, 66,190
710 DATA   2,  0, 64
```

```
720 DATA 255,255,255
730 DATA   1,255,128
740 DATA   1,189,128
750 DATA   3, 60,192
760 DATA  15,  0,240
770 DATA  15,  0,240
```

Dieses Programm aktiviert alle 8 Sprites mit derselben Definition, die in den DATA-Zeilen ab Zeile 560 steht, durch den SREAD-Befehl in Zeile 170 nach A$ übertragen wird und dort als Definitionsstring für den SDEFINE-Befehl dient. Dieses SDEFINE befindet sich in einer Schleife, die nacheinander alle 8 Sprites (0-7) anspricht und jeweils Farbe, Modus und Startposition festlegt (Zeilen 220-260). Die Formeln in den Zeilen sollten Sie nicht stören, es sei denn, Sie wollen es genau wissen. Schauen Sie sich einfach auf dem Bildschirm an, was durch sie bewirkt wird.

Nachdem nun in dieser Schleife alle Sprites vorbereitet "in den Startlöchern stehen", bekommen Sie nun durch die folgende FOR...NEXT-Schleife der Zeilen 300-320 einen gemeinsamen Zielpunkt in der Mitte des Bildschirms, den Sie sofort mit der Geschwindigkeit 1 ansteuern. Die Geschwindigkeit wurde bereits beim einfachen SSET-Befehl der ersten Schleife festgelegt.

Während die 8 Sprites ihren Weg zum Mittelpunkt des Bildschirms fortsetzen, läuft das BASIC-Programm weiter. Um dies zu demonstrieren, wird in den Zeilen 340-370 gleichzeitig mit der Spritebewegung eine hübsche Figur auf dem Bildschirm gezeichnet. Dieser Programmteil demonstriert zudem noch einmal eine kleine Anwendung des DRAW-Befehls.

Damit das Programm jetzt aber erst fortfährt, wenn das letzte Sprite steht, wird in Zeile 390 auf Sprite 0 gewartet, das den weitesten Weg hatte. Erst dann wird Zeile 400 ausgeführt.

Nach einer kurzen Wartezeit werden dann in den Zeilen 470-490 alle Sprites ausgeschaltet.

Dieses Programm enthält alle bisher bekannten Spritefunktionen. Sie können hier beobachten, wie sie in ein Programm integriert werden und welche Aufgaben sie haben. Doch Sie werden sehen, daß Ihre Supergraphik noch mehr zu bieten hat.

2.11.3 Kollisionen und BASIC-Interrupt

Vielleicht erinnern Sie sich noch an den Vorspann des Sprite-Kapitels. Dort wurde erwähnt, daß Ihr Rechner die Kollision eines Sprites mit einem anderen oder mit dem Hintergrund registriert. In Supergraphik wurde nun die Möglichkeit geschaffen, aus BASIC heraus einfach solche Kollisionen zu überprüfen oder sogar automatisch zu registrieren. Hierzu sind wieder eine Reihe von Befehlen zuständig, die im folgenden erläutert werden.

IF# ... THEN ...

Befehl:	IF# par THEN ...
Beispiel:	IF# U,1 THEN ...
Parameter:	par : Secundärfunktion
Funktion :	Bedingtes Verzweigen unter definierten Bedingungen

---

Mit diesem Befehlskomplex (auch hier existieren wieder (insgesamt 8) Sekundärbefehle) wird Ihnen ein besonderes Extra in die Hände gelegt, das z.B. Ihre Spielprogramme sehr entlastet. Die Funktion dieses Grundbefehls entspricht der des normalen IF...THEN...-Kommandos, es wird also eine bedingte Verzweigung ausgeführt.

Unter Verzweigung ist im folgenden ein Sprung hinter THEN gemeint. Wird nicht verzweigt, so geht das Programm in der nächsten Zeile weiter. Mit diesem Befehl können Sie nämlich nun die Joystickpositionen an den Eingangsports feststellen. p stellt die Nummer des Eingangsportes (1 oder 2) an der rechten Seite Ihres Gerätes dar. Nun können Sie folgende Befehle für Entscheidungen verwenden:

IF# U p THEN ...:
(U für up) Es wird verzweigt, wenn ein angeschlossener Joystick in Port p nach oben gedrückt wird.

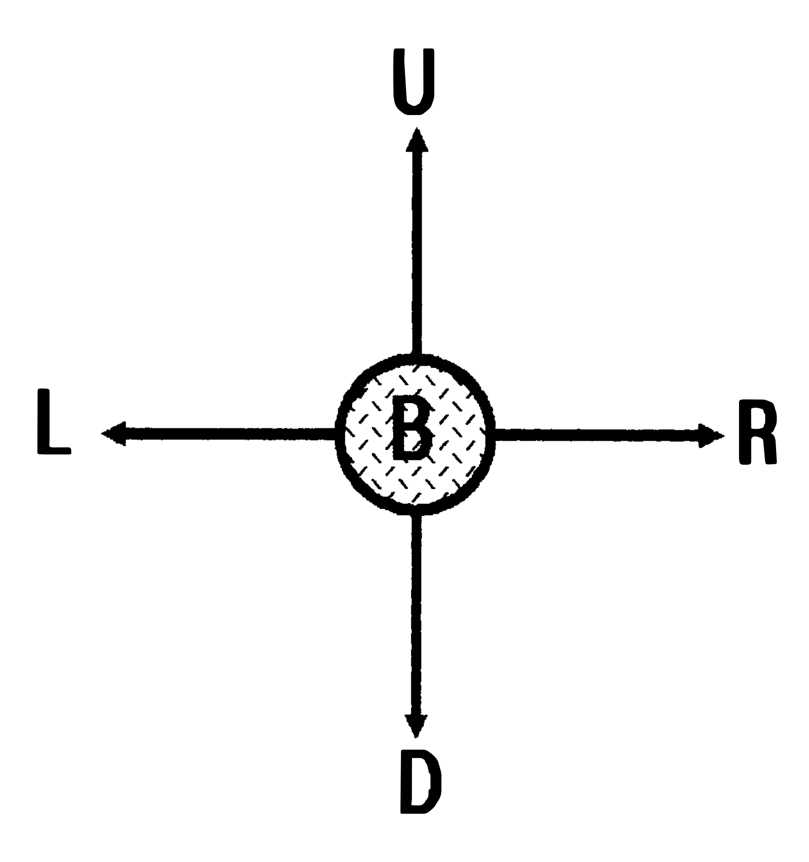
IF# D p THEN ...:
(D für down) Verzweigung bei Joystick nach unten.

IF# R p THEN ...:
Verzweigung bei Joystick nach rechts.

IF# L p THEN ...:
Verzweigung bei Joystick nach links.

IF# B p THEN ...:
Verzweigung, wenn der Feuerknopf (Button) Ihres Joysticks gedrückt wird.

```
10 IF# U2 THEN PRINT "OBEN"   : END
20 IF# D2 THEN PRINT "UNTEN"  : END
30 IF# R2 THEN PRINT "RECHTS" : END
40 IF# L2 THEN PRINT "LINKS"  : END
50 IF# B2 THEN PRINT "BUTTON" : END
60 GOTO 10
```

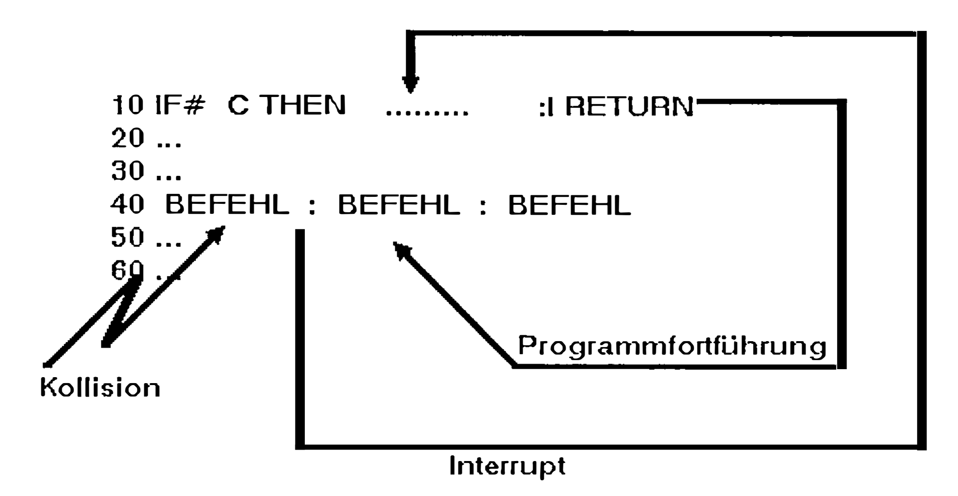


Dieses Programm erwartet an Port 2 die Joystickkommandos. Bei Port 1 ist die Überschneidung mit der Tastatur zu berücksichtigen!

Neben dieser Joystickabfrage können Sie aber auch die angekündigten Kollisionen zwischen Sprites etc. feststellen. Dafür gibt es zwei weitere Befehlsformen:

IF# CN THEN ...:
Verzweigung, wenn gerade im Augenblick eine Kollision eines Sprites mit einem anderen Sprite oder mit einem Hintergrundzeichen stattfindet.

IF# C THEN ...:
Nun wird es etwas komplizierter. Dieser Befehl ist nur etwas für diejenigen, die in BASIC schon etwas fester im Sattel sitzen. Trotzdem können Sie sich ihn ruhig einmal durchlesen, auch wenn Sie nicht alles verstehen sollten:



Im Prinzip handelt es sich hierbei um eine Art IRQ in BASIC, also ein Interrupt oder eine Unterbrechung in BASIC. Dieses Kommando wird z.B. am Anfang eines Programmes gegeben. Von nun an überprüft die Supergraphik 64 ständig nach jedem Befehl, ob eine Kollision eines Sprites mit einem anderen Sprite oder mit dem Hintergrund stattgefunden hat. Sollte dies der Fall sein, so merkt sich Ihr Computer die derzeitige Position im BASIC-Programm und führt einen Unterprogrammsprung hinter das THEN des gerade besprochenen

Befehls aus. Bei einem IRETURN (s.u.) wird dieses "Unterbrechungsprogramm" beendet und ganz normal an der Stelle weitergemacht, an der Ihr Programm sich gerade befand, als die Unterbrechung (durch eine Kollision verursacht) ausgeführt wurde.

In dieser Unterbrechungsroutine kann dann die Kollision abgearbeitet werden (z.B. ein Sprite explodieren lassen etc.), unter Beachtung der folgenden Dinge:

- In welcher Art von Kollision welche Sprites verwickelt wurden, steht in Registern 30 und 31 des Videocontrollers (s. Handbuch oder Hardwaregrundlagenkapitel). Jedes gesetzte Bit gibt dabei das Sprite an, das kollidierte. Diese Register werden bei IRETURN gelöscht.

- Während der Unterbrechungsroutine wird jede weitere Kollision ignoriert.

- Im Laufe der Unterbrechungsroutine müssen die Ursachen einer Kollision beseitigt werden, d.h. die beiden kollidierten Objekte müssen auseinander gebracht werden, da es ansonsten direkt nach der Beendigung der Unterbrechungsroutine wieder zu einer Unterbrechung kommen würde. Dies kann durch Umpositionieren oder Ausschalten eines der beiden Objekte geschehen.

- In Ihrer Unterbrechungsroutine sollten Sie keine Speicher des übrigen Programms verändern (außer, um besondere Effekte zu erzielen, wie im unten stehenden Beispielprogramm), da dies zu Konfusionen innerhalb Ihres Programms führen dürfte. Die Unterbrechung kann schließlich hinter jedem Befehl stattfinden.

- Hinter dem THEN des IF# C THEN-Befehls müssen vollständige Befehle stehen. Sie dürfen also nicht die sonst erlaubte Verkürzung:

```
IF# C THEN 520
```

verwenden, sondern müssen vollständig ausschreiben:

```
IF# C THEN GOTO 520
```

Dafür dürfen Sie aber den sonst notwendigen Doppelpunkt zwischen THEN und einem Supergraphik-Befehl weglassen.

- Der IRETURN-Befehl unterscheidet sich grundsätzlich vom RETURN-Befehl, um Unterprogramme zu beenden. Versuchen Sie also nie, ein Unterprogramm mit IRETURN oder die Unterbrechungsroutine mit RETURN abzuschließen.

- Weiterhin dürfen keine NEXT oder RETURN ohne vorherige FOR bzw. GOSUB in Ihrer Unterbrechungsroutine stehen. Nicht immer wird dadurch ein "RETURN WITHOUT GOSUB ERROR" oder "NEXT WITHOUT FOR ERROR" gegeben, da eventuell offene FOR ... NEXT - Schleifen oder Unterprogramme des übrigen Hauptprogrammes geschlossen werden. Achten Sie also besonders darauf!

- Sollte keine Unterprogrammebene mehr frei sein, so wird ein "FORMULA TOO COMPLEX ERROR" in der Zeile ausgegeben, die unterbrochen werden sollte.

Wenn Sie alle diese Punkte beherzigen, dann wird Ihnen dieser Befehl eine nützliche Hilfe sein.

IF# CC THEN ...:

Dieses Sekundärkommando hängt eng mit dem eben besprochenen C zusammen:

Es löscht den mit IF# C THEN ... gesetzten Unterbrechungsmodus und springt bei einer derzeitigen Kollision ein letztes Mal hinter das folgende THEN, d.h. von nun an wird die Unterbrechungsroutine bei einer Kollision nicht mehr aufgerufen.

IRETURN

Befehl:	IRETURN
Beispiel:	IRETURN
Parameter:	///
Funktion:	Beenden der Unterbrechungsroutine

Mit diesem Befehl schließen Sie eine Unterbrechungsroutine ab (s. IF# C THEN ...).

Hier ein kleines Beispielprogramm:

```
100 REM ********************
110 REM **                **
120 REM **  KOLLISIONEN   **
130 REM **                **
140 REM ********************
150 REM
160 GMODE 0,1 : GCLEAR : COLOR= 0,0
170 IF# C THEN GOTO 520 :REM INTERRUPTMODUS EIN
180 K=0                 :REM KOLLISIONSFLAG
190 REM
200 REM
210 REM 2 SPRITES VORBEREITEN:
220 REM
230 SREAD A$
240 SMODE 0,3,1 : SMODE 1,3,7
250 SSET 0,30,30,1 : SSET 1,200,200,1
260 SDEFINE 0,A$ : SDEFINE 1,A$
270 TEXT ,"BITTE TASTE BETAETIGEN!",80,20,0
280 WAIT 198,255 : POKE 198,0
290 REM
300 REM
310 REM SPRITES IN BEWEGUNG SETZEN:
320 REM
330 SSET 1 TO 30,30 : SSET 0 TO 200,200
340 REM
350 REM
360 REM WARTEN, BIS KOLLISIONSROUTINE
370 REM SPEICHER K AUF 1 SETZT:
380 REM
390 TEXT  ,"WARTEN!",200,90,0
400 TEXT 1,"WARTEN!",200,90,0
410 IF K=0 THEN 390
420 REM
430 TEXT ,"NOCHMAL(J/N)?",200,80,0
440 GET T$ : IF T$="" THEN 440
450 IF T$="J" THEN RUN
460 IF T$="N" THEN END
470 GOTO 440
480 REM
490 REM
500 REM KOLLISIONSROUTINE:
510 REM
520 COLOR= 1,1
530 FOR U=1 TO 100 : NEXT U
540 COLOR= 0,0
550 TEXT ,"CRASH!",50,50,0
560 SDEFINE 0,"" :REM KOLLISIONSURSACHE BESEITIGEN
570 K=1          :REM KOLLISIONSFLAG ZURUECKGEBEN
```

```
580 IRETURN
590 REM
600 REM
610 REM SPRITEDEFINITION:
620 REM
630 DATA   0,  0,  0
640 DATA   0,  0,  0
650 DATA   0,  0,  0
660 DATA   2,  0, 64
670 DATA   1,  0,128
680 DATA   0,129,  0
690 DATA   0, 66,  0
700 DATA   0, 60,  0
710 DATA   0,126,  0
720 DATA   0,195,  0
730 DATA   1,141,128
740 DATA   3, 44,192
750 DATA  31,255,248
760 DATA  62,153,124
770 DATA 125, 66,190
780 DATA   2,  0, 64
790 DATA 255,255,255
800 DATA   1,255,128
810 DATA   1,189,128
820 DATA   3, 60,192
830 DATA  15,  0,240
840 DATA  15,  0,240
```

Dieses kleine Demonstrationsprogramm setzt Ihnen zwei Sprites (ein gelbes und ein weißes) auf unterschiedliche Positionen des Bildschirms.

In Zeile 280 wartet das Programm dann auf einen Tastendruck Ihrerseits, um die beiden Sprites auf Kollisionskurs zu bewegen. Erfolgt die Kollision, so wird irgendwo innerhalb der Zeilen 390-410 unterbrochen. Durch das Einschalten des Interruptmodus in Zeile 170 wird die Kollisionsroutine in den Zeilen 520-580 aufgerufen, der Bildschirm blitzt auf, eine Meldung wird auf den Bildschirm geschrieben, Sprite 0 ausgeschaltet (um die Kollisionsursache zu beseitigen) und zurück zu dem Befehl gesprungen, der durch die Kollision unterbrochen wurde (innerhalb der Zeilen 390-410) gesprungen. Hier kann nun auf das Flag K reagiert werden, das die Kollisionsroutine auf 1 gesetzt hat.

Sprite 1 wird von der Kollisionsroutine nicht ausgeschaltet. Es wandert weiter und wird, wenn Sie nicht vorher auf eine Taste drücken, irgendwann mit dem Hintergrund in Form von Schrift zusammenstoßen. Dabei wird natürlich die Kollisionsroutine wieder aufgerufen.

Na, Demonstration genug? Bitte, jetzt sind Sie dran!

2.11.4 16 Sprites - Nicht nur Illusion

Wir haben bisher stets mit dem normalen, 8 Sprites umfassenden Spritesatz Ihres Commodore-Computers gearbeitet. Normalerweise ist dies auch die größte Anzahl an Sprites, die Sie ohne Probleme gleichzeitig auf dem Bildschirm darstellen können.

Mithilfe der erwähnten Interrupttechnik können Sie jedoch ein mehrfaches an Srites zur gleichen Zeit auf dem Bildschirm darstellen. Wir haben es hier mit einer ähnlichen Technik zu tun, die bereits angewandt wurde, um gleichzeitig Text und Graphik darzustellen (wir werden auf die genauen Prozesse im Hardwaregrundlagenkapitel noch genauer eingehen und Beispiele geben).

In der Supergraphik wurde nun ein Modus eingeführt, in dem es Ihnen möglich ist, insgesamt 16 verschiedene, völlig unabhängige Sprites gleichzeitig auf einem Bildschirm darzustellen. Dabei sind zwar einige Einschränkungen hinzunehmen, doch kann man damit sicher leben.

Zuständig für diesen zusätzlichen Spritesatz, sind die folgenden Befehle:

SPOWER

	Befehl:	SPOWER or,ur
Beispiel:	SPOWER 100,150	
Parameter:	or:	oberer Fensterrand
	ur:	unter Fensterrand
Funktion:	Zuschalten des zweiten Spritesatzes	

Hier ist er! So unscheinbar er aussieht, so Enormes bewirkt er. Mit diesem Befehl schalten Sie die evtl. vorher definierten (aber auch beliebig zuschaltbaren) Sprites mit den Nummern 8-15 ein. Dafür

definieren Sie (ähnlich dem GMODE 0,7,...-Befehl) ein Fenster, in dem diese weiteren 8 Sprites erscheinen sollen. In diesem Fenster können dafür jedoch keine Sprites mit niedrigeren Nummern mehr auftauchen. D.h. Sie legen auf Ihrem Bildschirm zwei Bereiche an, in denen die beiden Spritesätze getrennt auftauchen werden. Die zusätzlichen Sprites mit den Nummern 8-15 können Sie mit den ganz normalen Sprite-Befehlen, die wir bereits kennengelernt haben, bearbeiten - auch wenn der zweite Spritesatz noch nicht eingeschaltet ist.

Beim Einschalten dieses Befehls wird ein eventuell eingeschaltetes Textfenster in der Graphik automatisch abgeschaltet, da die beiden Zustände nicht gleichzeitig erscheinen können. Zu or und ur schlagen Sie bitte die Erläuterungen unter GMODE nach.

Sollte ein Sprite einmal ein wenig über die Fenstergrenze hinausreichen, so wird es trotzdem weitergezeichnet, wobei sich allerdings die Farbe etc. ändern kann. Außer der Definition nimmt der Teil des Sprites, der unterhalb des Fensters liegt, völlig die Eigenschaften des korrespondierenden Sprites des anderen Satzes an (unter korrespondierendem Sprite versteht man jeweils zwei Sprites aus den beiden Sätzen, deren Nummer gleich ist, wenn man von der Nummer des Sprites des zweiten Satzes 8 abzieht; also z.B.: Sprites 0 und 8 oder 7 und 15). So kann es vorkommen, daß Sie plötzlich zweifarbige hochauflösende Sprites auf dem Bildschirm zu sehen bekommen, was natürlich von Ihnen ausgenutzt werden kann.

Befehl:	SPOWER
Beispiel:	SPOWER
Parameter:	---
Funktion:	Abschalten des zweiten Spritesatzes

Wenn wir die zwei Parameter des SPOWER-Befehls weglassen, so wird das zuerst definierte Fenster umd damit die Sprites des zweiten Satzes ausgeschaltet. Dieser Befehl sollte stets entweder zu Beginn oder am Ende eines Programmes stehen, das zwischendurch einmal 16 Sprites einschaltet.

Im folgenden werden wir Ihnen ein kleines Programm vorstellen, das Ihnen 16 Sprites auf den Bildschirm zaubert und anschließend ein paar Sprites bewegt:

```
100 REM ******************
110 REM **              **
120 REM **  16 SPRITES  **
130 REM **              **
140 REM ******************
150 REM
160 GMODE 0,1 : GCLEAR : COLOR= 0,0
170 REM
180 FRAME 2,30,60,50 TO 260,150
190 TEXT ,"16 SPRITES",120, 95,0
200 TEXT ,"MIT",150,104,0
210 TEXT ,"SUPERGRAPHIK",113,113,0
220 SREAD S1$ : SREAD S2$
230 REM
240 REM
250 REM 16 SPRITES VORBEREITEN:
260 REM
270 FOR S=0 TO 15 STEP 2
280   SMODE S  ,0,S/2+1
290   SMODE S+1,3,15-S/2
300   X=S*30+40
310   IF S<8 THEN : SSET S,X,180 : SSET S+1,X,200
320   IF S>7 THEN : SSET S,X, 60 : SSET S+1,X, 90
330   SDEFINE S  ,S1$
340   SDEFINE S+1,S2$
350 NEXT S
360 REM
370 REM
380 REM 16 SPRITES AKTIVIEREN:
390 REM
400 SPOWER 10,100
410 REM
420 FOR X=1 TO 4000 : NEXT X
430 REM
440 REM
450 REM SPRITES BEWEGEN:
460 REM
470 SSET  7 TO 100,150,1
480 SSET  0 TO 150,230,1
490 SSET  9 TO 150,100,1
500 SSET 10 TO 100, 80,1
510 REM
520 WAIT 198,255
530 SWAIT 0 : SWAIT 7 : SWAIT 9 : SWAIT 10
540 SPOWER            :REM 2. SPRITESATZ AUS
550 REM
560 REM
570 REM SPRITEDEFINITION 1:
580 REM
590 DATA   0,  0,  0
600 DATA   0,  0,  0
```

```
610 DATA   0,  0,  0
620 DATA   0,  0,  0
630 DATA   0,  0,  0
640 DATA   0,  0,  0
650 DATA   0,  0,  0
660 DATA  24,  0,  0
670 DATA  28,  0,  0
680 DATA  30,  0,  0
690 DATA  31,  1,224
700 DATA  31,131,176
710 DATA  31,255,248
720 DATA 125,182,223
730 DATA  31,255,240
740 DATA   0,  0,  0
750 DATA   0,  0,  0
760 DATA   0,  0,  0
770 DATA   0,  0,  0
780 DATA   0,  0,  0
790 DATA   0,  0,  0
800 DATA   0,  0,  0
810 REM
820 REM
830 REM SPRITEDEFINITION 2:
840 REM
850 DATA   0,  0,  0
860 DATA   0,  0,  0
870 DATA  12,  0, 96
880 DATA   6,  0,192
890 DATA   3,  1,128
900 DATA   0,195,  0
910 DATA   0, 60,  0
920 DATA   0,126,  0
930 DATA   0,195,  0
940 DATA  17,141,136
950 DATA  19, 44,200
960 DATA  31,255,248
970 DATA  62,170,188
980 DATA 125, 85,126
990 DATA 255,255,255
1000 DATA 193,255,131
1010 DATA 193,129,131
1020 DATA   3,  0,192
1030 DATA  15,  0,240
1040 DATA  15,  0,240
1050 DATA   0,  0,  0
```

Wie Sie sehen, werden hier insgesamt 16 Sprites aus 2 verschiedenen Definitionen (s. DATA-Zeilen) in verschiedenen Farben und Größen auf dem Graphikbildschirm dargestellt.

Bevor das Programm jedoch irgendetwas in Sachen Sprites unternimmt, wird in den Zeilen 180-210 eine kleine Graphik gezeichnet, die dem ganzen etwas Form geben soll. Die Befehle hierzu sollten Ihnen inzwischen bekannt vorkommen.

Zeile 220 liest die beiden verschiedenen DATA-Definitionen in die Strings S1$ und S2$. Erst jetzt werden die 16 Sprites vorbereitet:

Dies übernimmt die FOR...NEXT-Schleife in den Zeilen 270-350. Hier werden z.B. die Farbe und die Größe unserer zukünftigen Sprites festgelegt (Zeilen 280/290). Dabei wird jedes zweite Sprite eine andere Größe und Form haben. Die Zeilen 310 und 320 positionieren die 16 Sprites, die schließlich in den Zeilen 330 und 340 eingeschaltet werden.

In diesem Zustand sind die oberen 8 Sprites zwar offiziell schon eingeschaltet, sie sind jedoch noch nicht auf dem Bildschirm zu sehen. Dies realisiert erst die Zeile 400 mit dem besagten SPOWER-Befehl. Hier wird zwischen den Bildschirmrasterzeilen 10 und 100 (nicht identisch mit den Graphikkoordinaten! s. GMODE) eine Zone eingerichtet, in der nur der obere Spritesatz gilt.

Es sind 16 Sprites gleichzeitig sichtbar!

Nach einer kurzen Wartezeit geschieht das unfaßbare: 4 der Sprites aus beiden Spritesätzen lösen sich aus der Formation und wandern über den Bildschirm (Zeilen 470-500). Und das, während das Programm weiterläuft!

Das Programm wartet auf einen Tastendruck, dann auf die 4 Sprites und knipst nun in Zeile 540 den zweiten Spritesatz aus, bevor es endet.

Während der Spritebewegung kann es zu einem kleinen Flackern der Sprites kommen. Dies liegt daran, daß sowohl die Spritebewegung als auch die Darstellung der 16 Sprites und die Tastaturabfrage über die gleiche Interruptroutine läuft, was manchmal, aufgrund des sehr zeitkritischen Verhaltens dieser Abläufe, zu Kollisionen führen kann.

Damit verlassen wir - ich hoffe nicht Sie - das große Kapitel der Sprites und wenden uns wieder anderen Problemen zu.

2.12 Wir bringen Farbe 'rein!

Bisher haben wir uns lediglich mit der Erzeugung der verschiedensten Formen und Gebilde herumgeschlagen. Die Farbe spielte dabei immer eine untergeordnete Rolle. Zwar haben wir auch dort bereits ein wenig auf die farbliche Gesamtgestaltung geachtet, ja sogar einige Farbbefehle kennengelernt, im großen und ganzen jedoch blieben uns die mannigfaltigen Möglichkeiten Ihres Rechners, die uns die Supergraphik erschließt, im unklaren. Dieses Versäumnis wollen wir an dieser Stelle nachholen.

2.12.1 Multicolor und Farbbefehle

Am Anfang unseres Exkurses in die Welt der Farben stehen die drei zentralen Farbbefehle. Sie werden nachher, wie wir sehen werden noch von einem allgemeinen Sekundärbefehl ergänzt, der bei allen Zeichenbefehlen angewandt werden kann, doch das soll uns hier und jetzt noch nicht stören.

Zunächst sollten wir uns aber über die normale Farbpalette unseres Commodore 64 klar werden. Aus diesem Grunde stellen wir im folgenden eine kleine Tabelle auf, die Sie - in erweiterter Form - auch im Anhang finden. Da Ihr Computer schlecht mit Farbnamen zurechtkommt, ist jeder der insgesamt 16 Farben, die der Rechner darstellen kann, eine Nummer von 0-15 zugeordnet:

hex	dez	Farbe	hex	dez	Farbe
$00	0	schwarz	$08	8	orange
$01	1	weiß	$09	9	braun
$02	2	rot	$0A	10	hellrot
$03	3	zyanblau	$0B	11	dunkelgrau
$04	4	violett	$0C	12	mittelgrau
$05	5	grün	$0D	13	hellgrün
$06	6	blau	$0E	14	hellblau
$07	7	gelb	$0F	15	hellgrau

Interessant für die Arbeit mit der Supergraphik sind natürlich nur die dezimalen Zahlenwerte. Werden Ihre Eingabewerte bei den verschiedensten Befehlen einmal größer als 15, so beginnt die Farbskala wieder von vorne, also 16=schwarz, 17=weiß usw. Der absolut größte Eingabewert ist 255; alle Werte, die größer sind als 255 erzeugen einen ILLEGAL QUANTITY ERROR.

Lange Rede, kurzer Sinn, fangen wir doch einfach einmal beim ersten Befehl an:

COLOR=

Befehl:	COLOR= r,h	
Beispiel:	COLOR= 2,6	
Parameter:	r:	Rahmenfarbe (0-15)
	h:	Hintergrundfarbe (0-15)
Funktion:	Festlegen der Rahmen- und Hintergrundfarbe	

Bei diesem Befehl, es ist der unkomplizierteste aller Farbbefehle, können Sie zunächst einmal mit dem Parameter r die Farbe des Rahmens Ihres Bildschirms festlegen. Dieser Rahmen bleibt bei allen Umschaltungen nach LGR, HGR, MC, Seiten 1 oder 2 solange in der gleichen Farbe, bis Sie ihn wieder mit diesem Befehl ändern.

Mit h dagegen wird die Hintergrundfarbe des jeweiligen Graphikfensters festgelegt. Sie kann für LGR (=Text), Seite 1 und Seite 2 (HGR/MC) jeweils verschiedene Werte annehmen. Beim Umschalten auf einen anderen Graphikmodus wird sie also mit berücksichtigt und dementsprechend umgeschaltet. D.h. Graphikseite 1 und 2 können beispielsweise jeweils andere Hintergrundfarben besitzen. Mit dem Parameter h wird also dasselbe bewirkt, wie mit einem SCOL= 0,h (s.u.).

Probieren Sie diesen Befehl doch einmal aus. Geben Sie im normalen Eingabemodus ein:

COLOR= 10,9 (return)

SCOL=

Befehl:	SCOL= n,f	
Beispiel:	SCOL= 1,14	
Parameter:	n:	Farbregister (0-3)
	f:	Farbe
Funktion:	Festlegen der Zeichenfarbe für den gesamten Bildschirm.	

SCOL= ist bereits ein recht komplexer Befehl. Um ihn richtig zu verstehen, sollten wir ein wenig über die Farbstruktur der Graphik wissen. Diejenigen, die es ganz genau wissen wollen, können natürlich die entsprechenden Kapitel in diesem Buch durchlesen, vielleicht blättern Sie ja auch noch einmal in dem kleinen Graphik-ABC am Anfang dieses großen 2. Kapitels. Hier soll nur ein kleiner Überblick gegeben werden:

Schauen wir uns die Verhältnisse doch einmal in der hochauflösenden Graphik an: In der hochauflösenden Graphik können wir - wie auch in den anderen Graphikmodi - gleichzeitig alle 16 versch7iedenen Farben auf dem Bildschirm darstellen. Dabei muß jedoch folgende Einschränkung gemacht werden: Pro 8x8-Punkte-Quadrat (also ein Feld in der Größe eines Buchstabens im Text-Modus) können wir nur zwei verschiedene Farben gleichzeitig auf dem Bildschirm erscheinen lassen. Dies ist zum einen die Hintergrundfarbe (die übrigens ebenfalls von 8x8-Feld zu 8x8-Feld unterschiedlich sein kann), also die Farbe für alle nicht gesetzten Punkte, zum anderen ist dies die Farbe der gesetzten Punkte in diesem 8x8-Feld. In einem anderen Quadrat können diese beiden Farben wieder völlig anders belegt sein.

Um diese Verhältnisse zu realisieren, besitzt Ihr Rechner einen separaten Farbspeicher, dessen Struktur wir hier nicht näher erläutern wollen (s. Kapitel Hardwargrundlagen). Wir nennen ihn auch Farbregister.

Von diesen Farbregistern gibt es nun in HGR genau zwei: eines für die Hintergrundfarbe (wir nennen es Farbregister 0), eines für die Farbe aller gesetzten Punkte (Farbregister 1). In diesen Farbregistern steht nun die Nummer der Farbe, die die gesetzten oder nicht gesetzten Punkte besitzen sollen.

Kommen wir nun zur Multicolorgraphik MC). Hier sieht das Ganze ein wenig komplizierter aus: In MC ist es Ihnen möglich, insgesamt 4 verschiedene Farben gleichzeitig in einem oben erwähnten Quadrat

unterzubringen. Da jedoch ein Punkt nun die doppelte Breite besitzt, ist ein solches Quadrat nur noch 4x8-Punkte groß. Entsprechend sinkt die Graphikauflösung bekanntlich auf 160x200 Punkte.

Auch hier existieren wieder verschiedene Farbregister, diesmal 4 an der Zahl (Farbregister 0-3). Im Unterschied zur hochauflösenden Graphik gilt in MC das Farbregister 0 für alle 4x8-Punkte-Kästchen auf dem gesamten Bildschirm. Die Register 1,2 und 3 dagegen können auch hier von Kästchen zu Kästchen variieren.

Damit Ihr Rechner nun weiß, ob er die Farbe eines Punktes in MC aus dem Register 0, 1, 2 oder 3 beziehen soll, wird nicht gespeichert, ob der Punkt gesetzt oder nicht gesetzt ist. Es wird vielmehr die Nummer des Registers (0-3) vermerkt, aus dem die Farbe stammen soll (da hierfür statt 1 insgesamt 2 Bits pro Punkt notwendig sind verringert sich - wie erwähnt - die Graphikauflösung, s. Hardwaregrundlagen).

Was macht nun der Befehl SCOL=?

Mit SCOL= setzen Sie die Farbregister 0-3 auf dem gesamten Bildschirm, d.h. der gesamte Bildschirm ist nun für das angegebene Farbregister einfarbig, die Farbregister jedes 8x8- bzw. 4x8-Feldes auf dem gesamten Bildschirm enthalten die gleiche Farbnummer (n=0 ist die Hintergrundfarbe; SCOL=0,f entspricht also COLOR=r,f). Gleichzeitig wird die entsprechende Punktfarbe gesetzt (s. PCOL=).

SCOL= wird gewöhnlich zur Initialisierung der Graphikfarben verwendet. Das folgende Beispiel arbeitet in Multicolorgraphik und soll die Funktion von SCOL= erläutern:

```
100 REM ********************************
110 REM **                            **
120 REM **  SCOL= - FARBEN, FARBEN    **
130 REM **                            **
140 REM ********************************
150 REM
160 GMODE 0,2 : GCLEAR
170 SCOL= 0,0 : SCOL= 1,2
180 SCOL= 2,4 : SCOL= 3,6
190 REM
200 REM MIT REGISTER-NUMMER 1 ZEICHNEN:
210 REM
220 PCOL= 4,1
```

```
230 FILL , 30, 30 TO  60, 60
240 FRAME ,7, 90, 90 TO 120,120
250 TEXT ,"REGISTER 1",20,160,0
260 REM
270 REM MIT REGISTER-NUMMER 2 ZEICHNEN:
280 REM
290 PCOL= 4,2
300 FILL , 40, 40 TO  70, 70
310 FRAME ,7,100,100 TO 130,130
320 TEXT ,"REGISTER 2",20,168,0
330 REM
340 REM MIT REGISTER-NUMMER 3 ZEICHNEN:
350 REM
360 PCOL= 4,3
370 FILL , 50, 50 TO  80, 80
380 FRAME ,7,110,110 TO 140,140
390 TEXT ,"REGISTER 3",20,176,0
400 REM
410 REM FARBEN WECHSELN:
420 REM
430 FOR F=0 TO 100
440   FOR R=1 TO 3
450     SCOL= R,F+3*R
460     FOR T=1 TO 200 : NEXT T
470   NEXT R
480 NEXT F
```

Schauen wir uns doch einmal etwas genauer an, was in diesem Programm geschieht:
Initialisiert wird unser Programm in den Zeilen 160-180. Hier wird zunächst mit einem

```
GMODE 0,2
```

die Multicolorgraphik Seite 1 eingeschaltet, um dann mit dem bekannten

```
GCLEAR
```

gelöscht zu werden. Im Anschluß daran (Zeile 170) wird zunächst einmal die Hintergrundfarbe auf schwarz gesetzt:

```
SCOL= 0,0
```

Wie Sie bereits wissen, ist die Hintergrundfarbe für das gesamte Graphikbild identisch. Als nächstes werden nacheinander Farben 1 bis 3 für den gesamten Bildschirm gesetzt:

```
SCOL= 1,2
SCOL= 2,4
SCOL= 3,6
```

Damit hätten wir alle Farbregister zum Start vorbereitet, und es kann losgehen.

In den Zeilen 220, 290 und 360 finden Sie jeweils einen Befehl, der Ihnen noch unbekannt ist (er wird gleich etwas weiter unten beschrieben), Sie aber nicht weiter stören soll. Er hat in diesem Fall lediglich die Aufgabe, der Supergraphik anzugeben, mit welcher Registernummer in Zukunft gezeichnet werden soll. Damit schaltet das Programm also in Zeile 220 auf Registernummer 1. In den Zeilen 230-250 werden also stets mit Registernummer 1 nacheinander ein Feld mit FILL, ein Rahmen mit FRAME und ein Schriftzug mit TEXT gezeichnet. In Zeile 290 wird dann auf Registernummer 2 umgeschaltet und der gleiche Vorgang wiederholt sich. Die Zeilen 360-390 bewirken analoges mit Registernummer 3.

Jetzt sind also drei mal drei Gebilde mit allen drei Registernummern, also mit drei verschiedenen Farben, auf den Bildschirm gezeichnet worden. Mit dem Befehl SCOL= können wir nun die Farbzuordnungen zu den verschiedenen Registern ändern, so daß die Farbe der verschiedenen Gebilde schlagartig wechselt, ohne daß sie neu gezeichnet werden müssen. Dies wird dann auch in den Zeilen 430-480 realisiert. Hier ändern zwei verschachtelte FOR...NEXT-Schleifen jeweils die Farbregisternummer (innere Schleife) und die zuzuordnende Farbnummer (äußere Schleife).

Machen Sie sich keine Gedanken über den Wertebereich des Farbnummernparameters. Wie Sie oben bereits lesen konnten, gibt es zwar nur 16 Farben, für die die Nummern 0-15 reserviert sind, höhere Werte können trotzdem eingegeben werden, da Supergraphik diese Werte wieder heruntertransformiert. Bei Nummer 16 fängt die Farbskala also wieder an.

Vielleicht sollten Sie ein wenig mit diesem Befehl herumprobieren, ihn auch einmal in HGR anwenden usw. Wenn Sie mit ihm etwas vertraut sind, fällt die Anwendung dann auch nicht mehr allzu schwer.

PCOL=

Befehl:	PCOL= n,f
Beispiel:	PCOL= 1,14
Parameter:	n: Farbregisternummer/Registermodus (0-4)
	f: Farbnummer (0-15)/Farbregisternr. (0-3)
Funktion:	Festlegen der Zeichenfarbe

Im obigen SCOL=-Beispiel haben wir ihn bereits einmal kennengelernt. Dieser Befehl hat insgesamt 2 verschiedene Aufgaben. Die erste Aufgabe nimmt er war, wenn der erste Parameter n auf 4 gesetzt wird und ist nur für MC interessant. In dieser Funktion haben wir ihn bereits kennengelernt. Hier bestimmt der zweite Parameter dann, mit welcher Registernummer in Multicolor gezeichnet werden soll.

PCOL= 4,1

bestimmt dann beispielsweise, daß alle Figuren, Texte oder Punkte von nun an mit der Registernummer 1 gezeichnet werden, also die Farbe des Farbregisters 1 annehmen.

Bewegt sich der erste Parameter dagegen im Bereich 0-3 (in HGR natürlich nur von 0-1), so gilt folgendes:

Der erste Parameter n bestimmt das Farbregister. Der zweite Parameter gibt dann die Farbe an, die dem Register zugeordnet wird. Insoweit entspricht in diesem Fall PCOL= dem Befehl SCOL=. Bei SCOL= wird nun aber auch der gesamte Bildschirm sofort mit dieser Farbe eingefärbt, d.h. alle Farbregister 1 des Bildschirms (bekanntlich hat jeder 4x8- bzw. 8x8-Block der Graphik ein eigenes Farbregister 1) werden mit der im zweiten Parameter angegebenen Farbe belegt.

Bei PCOL= wird dagegen nur der Speicher für das aktuelle Farbregister geändert. Auf dem Bildschirm tut sich noch gar nichts. Erst wenn Sie ein Gebilde zeichnen, erhält dieses Gebilde die angewählte Farbe.

In der Tat wird beim SCOL=-Befehl ebenfalls der Speicher für das aktuelle Farbregister gesetzt, zusätzlich aber - wie gesagt - der gesamte Bildschirm mit dieser Farbe eingefärbt.

Doch so einfach geht die Sache doch nicht. Denn normalerweise werden bei allen Zeichenbefehlen wie PLOT, CIRCLE etc. keine Farben gesetzt. Es wird nur in HGR gezeichnet oder gelöscht und in

MC mit den verschiedenen Farbregistern gezeichnet. Es wird jedoch noch nicht berücksichtigt, daß ja jedes 4x8- bzw. 8x8-Feld eine eigene Farbzuordnung für seine Farbregister haben kann. Hier wird nun ein neuer Sekundärbefehls-Zusatz wichtig, den wir bisher noch nicht besprochen haben:

Die C-Erweiterung:

Sicher erinnern Sie sich noch an die beiden Erweiterungen aller Zeichenbefehle: T für Test und B für Pinselwahl.

C ist nun ein weiterer, letzter Zusatz, der unabhängig von und zusätzlich zu den beiden anderen an einen Zeichenbefehl angehängt werden kann. Dabei ist bei der Kombination mit den beiden anderen Zusätzen lediglich zu beachten, daß der C-Zusatz an erster Stelle kommt, also z.B.:

```
PLOT C B 7, T TEST,1,100,150
```

Damit ergeben sich wieder eine Reihe anderer Kombinationsmöglichkeiten mit den ganz normalen Zeichenbefehlen:

```
PLOT                 (zm),x,y
PLOT C               (zm),x,y
PLOT   B m,          (zm),x,y
PLOT C B m,          (zm),x,y
PLOT         T var,  (zm),x,y
PLOT C       T var,  (zm),x,y
PLOT   B m,  T var,  (zm),x,y
PLOT C B m,  T var,  (zm),x,y
```

Doch was bewirkt dieser neue kleine Buchstabe nun eigentlich?
Wie gesagt, wird durch ihn bewirkt, daß die einem Farbregister zugeordnete Farbe aus dem internen aktuellen Speicher der Supergraphik auch in die Farbregister derjenigen 4x8- bzw. 8x8-Felder übertragen wird, die von der gezeichneten Figur (Linie, Punkt,...) angesprochen werden. Es wird also nicht nur vermerkt, aus welchem Farbregister die Linie etc. ihre Farbe beziehen soll (was ja auch ohne C-Zusatz geschieht), sondern welche Farbe den angesprochenen Farbregistern nun zugeordnet werden soll. Während also durch den Befehl SCOL= alle Farbregister des gesamten Bildes einheitlich mit einer Farbe besetzt werden, können Sie mit dem C-Zusatz (gemeinsam mit PCOL=) hier noch weiter differenzieren.

Ein Beispiel mag die Notwendigkeit dieses Zusatzes erläutern:
Nehmen Sie einmal an, Sie würden zwei parallele, nahe beieinander liegende Linien mit Farbsetzung zeichnen; etwa mit folgenden Befehlen:

```
10 GMODE 0,1 : GCLEAR
20 PCOL= 1,2 : PLOT C ,0,0 TO 159,199
30 PCOL= 1,3 : PLOT C ,0,3 TO 156,199
40 WAIT 198,255
```

Haben Sie dieses kleine Programm gestartet, werden Sie bemerken, daß beide Linien (hier in HGR) die Farbe der zuletzt gezeichneten annehmen, obwohl beide in verschiedenen Farben gezeichnet werden sollten (die erste in rot, die zweite in zyanblau). Dies resultiert aus dem oben erläuterten Problem der Farbauflösung (in HGR nur 2 Farben in einem 8x8-Feld).

Wollen Sie nun aber die Farbe der zuerst gezeichneten Linie erhalten, so setzen Sie lediglich hinter das PLOT der ersten Linie (also in Zeile 20) ein C, die zweite Linie wird dann ohne Farbsetzung gezeichnet (was übrigens auch ein wenig schneller vonstatten geht).

Wenn Sie generell ohne den C-Zusatz zeichnen, wie wir das bisher immer getan haben, so wird stets in den Farben gezeichnet, die Sie mit dem SCOL=-Befehl gesetzt haben.

Im LGR-Modus, also in der niedrig auflösenden Graphik, die im normalen Text-Fenster realisiert wird, bewirkt der C-Zusatz lediglich, daß nun nicht - wie sonst - die aktuelle Textfarbe als Punktfarbe angesehen, sondern die reguläre Punktfarbe 1 (mit PCOL= 1,f bestimmt) verwendet wird.

Lediglich im Zusammenhang mit Zeichenmodus zm=4 hat C keine besondere Wirkung. Im Zusammenhang mit zm=1 muß darauf geachtet werden, daß ebenfalls die aktuelle PCOLor gesetzt wird. Wollen Sie also mit einer bestimmten anderen Farbe löschen, so verwenden Sie den PCOL= zur Änderung der Punktfarbe.

Das folgende Programm soll Ihnen den Gebrauch von PCOL= und C ein wenig näher bringen:

```
100 REM **********************************
110 REM **                              **
120 REM **  PCOL= - NOCH MEHR FARBEN    **
130 REM **                              **
140 REM **********************************
150 REM
160 GMODE 0,2 : GCLEAR
170 SCOL= 0,0 : SCOL= 1,6
180 SCOL= 2,2 : SCOL= 3,5
190 REM
200 REM ***************
210 REM **           **
220 REM **  FIGUR 1  **
230 REM **           **
240 REM ***************
250 REM
260 R=1 :REM REGISTERNUMMER
270 FOR X=0 TO 79 STEP 4
280   R=R+1 : IF R=4 THEN R=1
290   PCOL= 4,R
300   CIRCLE ,, 80,100,X,99-X/2
310   CIRCLE ,, 79,100,X,99-X/2
320 NEXT X
330 REM
340 FOR W=1 TO 3000 : NEXT W
350 REM
360 REM ***************
370 REM **           **
380 REM **  FIGUR 2  **
390 REM **           **
400 REM ***************
410 REM
420 GCLEAR
430 PCOL= 4,1 : PCOL= 1,2
440 TEXT C ,"FARBENSPIEL",37,10,0
450 R=1 : F=1
460 FOR X=0 TO 380 STEP 3
470   R=R+1 : IF R= 4 THEN R=1
480   F=F+1 : IF F=16 THEN F=1
490   PCOL= 4,R
500   PCOL= R,F
510   S1=50*SIN(X/30)+100
520   S2=30*SIN(X/20)+80
530   FILL C 0,S2,S1 TO S2+2,S1+2
540 NEXT X
550 WAIT 198,255
```

Auch in diesem Programm werden zunächst einmal in den Zeilen 160-180 die Graphik und alle Farbregister initialisiert (auch hier befinden wir uns wieder in Multicolor).

In Zeile 260 geht es dann richtig los. Wir wollen hier nicht erläutern, wie die hier gezeichnete Figur realisiert wird, das können Sie leicht anhand des Listings und ein paar Versuchen zur Programmänderung selbst feststellen. Wesentlich soll an diesem Programm ja der Umgang mit der Farbgebung sein.

So wird in der FOR...NEXT-Schleife der Figur 1 die Variable R (für Registernummer) ständig von 1 bis 3 hochgezählt, um nach und nach in allen 3 Farbregistern (ohne Register 0 als Hintergrundfarbe natürlich) die verschiedenen Kreise zu zeichnen. Da die Kreise ohne Farbsetzung gezeichnet werden, erscheinen Sie lediglich in den drei durch die SCOL=-Initialisierung vorgegebenen Farben auf dem Bildschirm.

Figur 2 nun demonstriert Ihnen, daß es auch möglich ist, alle 16 Farben gleichzeitig auf dem Bildschirm darzustellen. Hierzu wird in Zeile 470 stets die Registernummer von 1-3, in Zeile 480 die Farbnummer von 1-16 gezeichnet.

Sie sollten hier ein wenig experimentieren, um sicher zu werden. Das nächste Kapitel zeigt Ihnen eine sehr schöne, bereits weiterreichende Anwendung der Farbgebung in der Supergraphik.

2.12.2 Nur 16 Farben? Farbmischung! 136 Farben für den Commodore 64

Man sollte meinen, 16 Farben reichen für die meisten Anwendungen. Doch jeder Computermensch strebt nach neuen, immer höheren Leistungsmerkmalen und so werden wir hier eine Möglichkeit vorstellen, mit der wir gleichzeitig sehr viel mehr als nur die üblichen 16 Farben auf dem Bildschirm darstellen können. Es gibt natürlich auch Anwendungen, bei denen es darauf ankommt, möglichst viele verschiedene Farben und besonders Farbschattierungen auf dem Bildschirm darzustellen. Man denke hierbei allein einmal an dreidimensionale Objekte, auf denen der Lichteinfall durch unterschiedliche Schattierungen dargestellt werden kann.

Die Lösung für dieses Vorhaben auf unserem Commodore 64 liegt in der Mischung verschiedener Farben. Bekanntlich hat das Auge nur eine beschränkte Punktauflösung. Zwei sehr nahe beieinander liegende

Punkte erscheinen deshalb als ein einziger. Haben diese beiden Punkte nun unterschiedliche Farben, so registriert das Auge eine Mischung aus beiden Farben. Man unterscheidet dabei die sogenannte additive oder die subtraktive Farbmischung. Die subtraktive Farbmischung kommt bei der Mischung von z.B. Wasserfarben zustande, da durch die Mischung zweier Farben hier insgesamt weniger Licht zu Auge fällt (die beiden gemischten Farben behalten stets ihren Lichtanteil bei sich). Bei Ihrem Monitor oder allgemein bei der Mischung von Lichtquellen werden die beiden Farben addiert, es gelangt also mehr Licht ans Auge.

Diese additive Farbmischung wird beispielsweise beim normalen Farbfernseher ausgenutzt. Hier setzt sich das Bild aus den drei Grundfarben Rot, Grün und Blau (RGB) zusammen, die jeweils gemischt werden. Die Farbe Weiß entsteht durch das Zusammenwirken aller drei Farben in gleicher Intensität. Schwarz bedeutet das Fehlen jeglichen Lichtes. Soll z.B. die Farbe Rosa dargestellt werden, so werden die beiden Farben Grün und Blau in ihrer Intensität ein wenig gegenüber Rot zurückgenommen etc.

Diese Mittel stehen uns bei unserem Computer nicht direkt zur Verfügung, obwohl auch er sein Farbbild auf diese Weise erstellt. Wir müssen jedoch mit den zur Verfügung stehenden 16 Farben unseres Computers auskommen. Eine Farbmischung erreichen wir durch Untereinandersetzen zweier gleich- oder verschieden farbiger Punkte in Multicolor. Ein einziger Punkt setzt sich damit aus zwei Multicolor-Punkten zusammen.

Mit dieser Art der Farbmischung lassen sich dann eine ganze Menge mehr Farben auf dem Bildschirm darstellen, als bisher. Alle 16 Farben können wir mit wiederum 16 Farben kombinieren und kämen dann rein rechnerisch auf 16^2=256 verschiedene Kombinationen. Da es jedoch zum selben Ergebnis führt, wenn wir schwarz mit weiß oder weiß mit schwarz kombinieren, fallen eine ganze Reihe Kombinationen weg. Es bleiben insgesamt 136 verschiedene sinnvolle und auch recht schöne Kombinationen. Davon sind genau 120 Farben neu und das Produkt der eben besprochenen Farbmischung.

Wie dieses Vorhaben nun mit unserer Supergraphik realisiert wird, zeigt uns das folgende BASIC-Programm, das sämtliche 136 Farben in kleinen Rechtecken auf den Bildschirm bringt:

```
100 REM ************************
110 REM **                    **
120 REM **  VIEL MEHR FARBEN  **
130 REM **                    **
140 REM ************************
150 REM
160 GMODE 0,2 : GCLEAR
170 SCOL= 0,0
180 SCOL= 1,2 : SCOL= 2,3 : SCOL=3,9
190 REM
200 REM
210 REM FARBKOMBINATIONEN UEBERNEHMEN:
220 REM
230 DIM FS(136,1)
240 FA=0
250 FOR X=0 TO 15      :REM ERSTFARBE
260   FOR Y=X TO 15    :REM ZWEITFARBE
270     READ FS(FA,0),FS(FA,1)
280     FA=FA+1        :REM FARBNUMMER+1
290   NEXT Y
300 NEXT X
310 REM
320 REM
330 REM RAHMEN, UEBERSCHRIFT:
340 REM
350 PCOL= 4,3  :REM REGISTER 3 EIN
360 FRAME 2,3,20,18 TO 159,158
370 PCOL= 2,13 :REM FARBE 13 NACH REG. 2
380 PCOL= 4,2  :REM REGISTER 2 EIN
390 TEXT C ,"FARBSKALA",50,10,0
400 REM
410 REM
420 REM FARBFELDER ZEICHNEN:
430 REM
440 F=0
450 FOR X=0 TO 15      :REM ERSTFARBE
460   FOR Y=X TO 15    :REM ZWEITFARBE
470     X1=X*8+24 : X2=X*8+7+24
480     Y1=Y*8+24 : Y2=Y*8+7+24
490     FA=F
500     GOSUB 800      :REM FELD ZEICHNEN
510     F=F+1          :REM FARBE ERHOEHEN
520   NEXT Y
530 NEXT X
540 REM
550 REM
560 REM BESCHRIFTUNG:
570 REM
580 PCOL= 2,8 : PCOL= 4,2
590 FOR X=0 TO 15
600   TEXT C ,STR$(Z),0,X*8+32,0   :REM EINER LINKS
```

```
610   TEXT C ,STR$(Z),X*8+16,176,0 :REM EINER UNTEN
620   Z=Z+1 : IF Z=10 THEN Z=0
630   IF X<10 THEN 660
640   TEXT C ,"1",0,X*8+32,0       :REM ZEHNER LINKS
650   TEXT C ,"1",X*8+24,168,0     :REM ZEHNER UNTEN
660 NEXT X
670 REM
680 WAIT 198,255
690 END
700 REM
710 REM
720 REM FELD ZEICHNEN:
730 REM UEBERGABE:
740 REM           FA- FARBE
750 REM           X1- X1-KOORD.
760 REM           Y1- Y1-KOORD.
770 REM           X2- X2-KOORD.
780 REM           Y2- Y2-KOORD.
790 REM
800 FOR Y0=Y1 TO Y2 STEP 2
810   PCOL= 1,FS(FA,0) : PCOL= 2,FS(FA,1)
820   PCOL= 4,3
830   PLOT C B 102,,X1,Y0   TO X2,Y0
840   PLOT C B 153,,X1,Y0+1 TO X2,Y0+1
850 NEXT Y0
860 RETURN
870 REM
880 REM
890 REM FARBKOMBINATIONSTABELLE:
900 REM
910 DATA  0, 0 ,  0, 1 ,  0, 2 ,  0, 3
920 DATA  0, 4 ,  0, 5 ,  0, 6 ,  0, 7
930 DATA  0, 8 ,  0, 9 ,  0,10 ,  0,11
940 DATA  0,12 ,  0,13 ,  0,14 ,  0,15
950 DATA          1, 1 ,  1, 2 ,  1, 3
960 DATA  1, 4 ,  1, 5 ,  1, 6 ,  1, 7
970 DATA  1, 8 ,  1, 9 ,  1,10 ,  1,11
980 DATA  1,12 ,  1,13 ,  1,14 ,  1,15
990 DATA                  2, 2 ,  2, 3
1000 DATA  2, 4 ,  2, 5 ,  2, 6 ,  2, 7
1010 DATA  2, 8 ,  2, 9 ,  2,10 ,  2,11
1020 DATA  2,12 ,  2,13 ,  2,14 ,  2,15
1030 DATA                          3, 3
1040 DATA  3, 4 ,  3, 5 ,  3, 6 ,  3, 7
1050 DATA  3, 8 ,  3, 9 ,  3,10 ,  3,11
1060 DATA  3,12 ,  3,13 ,  3,14 ,  3,15
1070 REM
1080 DATA  4, 4 ,  4, 5 ,  4, 6 ,  4, 7
1090 DATA  4, 8 ,  4, 9 ,  4,10 ,  4,11
1100 DATA  4,12 ,  4,13 ,  4,14 ,  4,15
1110 REM
```

```
1120 DATA          5, 5 ,  5, 6 ,  5, 7
1130 DATA  5, 8 ,  5, 9 ,  5,10 ,  5,11
1140 DATA  5,12 ,  5,13 ,  5,14 ,  5,15
1150 REM
1160 DATA                  6, 6 ,  6, 7
1170 DATA  6, 8 ,  6, 9 ,  6,10 ,  6,11
1180 DATA  6,12 ,  6,13 ,  6,14 ,  6,15
1190 REM
1200 DATA                          7, 7
1210 DATA  7, 8 ,  7, 9 ,  7,10 ,  7,11
1220 DATA  7,12 ,  7,13 ,  7,14 ,  7,15
1230 REM
1240 REM
1250 DATA  8, 8 ,  8, 9 ,  8,10 ,  8,11
1260 DATA  8,12 ,  8,13 ,  8,14 ,  8,15
1270 REM
1280 REM
1290 DATA          9, 9 ,  9,10 ,  9,11
1300 DATA  9,12 ,  9,13 ,  9,14 ,  9,15
1310 REM
1320 REM
1330 DATA                 10,10 , 10,11
1340 DATA 10,12 , 10,13 , 10,14 , 10,15
1350 REM
1360 REM
1370 DATA                         11,11
1380 DATA 11,12 , 11,13 , 11,14 , 11,15
1390 REM
1400 REM
1410 REM
1420 DATA 12,12 , 12,13 , 12,14 , 12,15
1430 REM
1440 REM
1450 REM
1460 DATA         13,13 , 13,14 , 13,15
1470 REM
1480 REM
1490 REM
1500 DATA                 14,14 , 14,15
1510 REM
1520 REM
1530 REM
1540 DATA                         15,15
```

Aus diesem Programm können Sie die genaue Technik ersehen, die der Erzeugung der Mischfarben dient. Fangen wir also einfach einmal an, das vorliegende Programm unvoreingenommen zu beschreiben:

Gestartet wird wieder mit der fast schon obligatorischen Initialisierung in den Zeilen 160-180. Als nächstes werden die möglichen Farbnummernkombinationen aus der am Ende des Programmes positionierten DATA-Zeilen-Tabelle in einen zwei-dimensionalen Array FS() übernommen. Damit ist es möglich, jede Kombination durch eine bestimmte Nummer FA aufzurufen. Die Übernahme geschieht in zwei verschachtelten FOR...NEXT-Schleife in den Zeilen 240-300.

Nach diesen Vorbereitungen, die ein wenig Programm-Ausführzeit in Anspruch nehmen, geht es nun zum Zeichnen eines Rahmens und einer passenden Überschrift in den Zeilen 350-390. Die hierzu vorgenommene Farbgebung sollten Sie nun bereits schon verstehen.

Die eigentliche Hauptarbeit wird dagegen in den Zeilen 440-530 vorgenommen. Hier werden in zwei verschachtelten FOR...NEXT-Schleifen (ähnlich wie bei der Tabellenübernahme oben) alle 136 Mischfarben nacheinander aufgerufen und nach ihren Grundfarben geordnet auf den Bildschirm gebracht. Zuständig für das Zeichnen eines Rechteckes aus jeweils zwei Mischfarben ist ein Unterprogramm, das bei Zeilennummer 800 beginnt.

In diesem Unterprogramm werden aus der übergebenen Farbnummer FA die beiden Mischfarben 1 und 2 ermittelt und in Zeile 810 mittels PCOL= gesetzt. Was nun kommt erscheint etwas kompliziert. Zunächst wird dafür gesorgt, daß in Farbregisternummer 3 gezeichnet wird (Zeile 820). Das hat aber nur nebensächlichen Charakter, denn tatsächlich wird nämlich gar nicht mit dieser Registernummer gezeichnet. Was geschieht denn nun wirklich?

Nun, es werden einfach erst einmal zwei direkt untereinander liegende Linien gezeichnet (Zeilen 830/840). Dies geschieht nun aber im B-Modus, d.h. mit Pinselwahl. Vielleicht erinnern sie sich noch, daß Sie nach dem B-Zusatz eine Zahl angeben konnten, die als Bitmuster an diejenige Speicherstelle übertragen wird, die sonst nur durch einen einzelnen Punkt angesprochen wird. Da wir aber in Multicolor jeweils 2 Bit pro Punkt im Graphikspeicher reserviert haben, um die Nummer der 4 verschiedenen Farbregister zu vermerken, hat diese Pinselwahl nun sogar Auswirkungen auf das Farbmuster des Pinsels. Sie brauchen dies nicht unbedingt zu verstehen, da hierzu die Kenntnis über den Aufbau des Graphikspeichers nötig wäre. Sie können sich einfach merken, daß durch den Pinsel Nr. 102 abwechselnd Punkte mit Registernummer 1 und 2, durch den Pinsel Nr. 153 dagegen abwechseln mit Registernummer 2 und 1 gezeichnet wird.

Für Interessierte:
Der dezimale Wert 102 ergibt binär den Wert

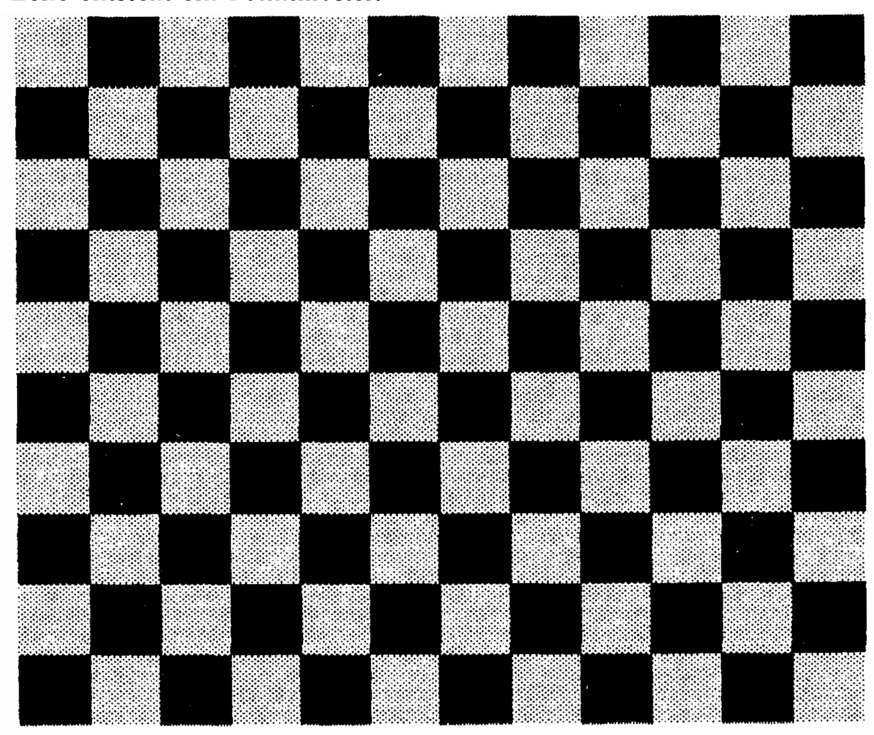
%01 10 01 10

der Wert 153 ist dagegen:

%10 01 10 01

Damit werden im Speicher abwechselnd die Kombinationen 01 und 10 für Registernummer 1 und 2 abgelegt.

Auf jeden Fall wird durch diesen Wechsel zwischen Register 1 und 2 eine gepunktete Linie erzeugt. Durch das Versetzen in der nächsten Zeile entsteht ein Punktmuster:



■ =Farbe 2

░ =Farbe 1

Damit ist dafür gesorgt, daß jeweils zwei verschieden farbige Punkte möglichst nahe beieinander liegen, gleichzeitig aber auch eine Streifenbildung verhindert wird.

Die umhüllende FOR...NEXT-Schleife erzeugt dann das vollständige Rechteck.

Bitte beachten Sie dabei aber folgendes: Es können hierbei stets nur Rechtecke gezeichnet werden, die aus vollständigen 4x8-Punktekästchen bestehen. Dies ist auch sinnvoll, da ansonsten Farbüberschneidungen resultieren würden, die von der Farborganisation herrühren.

Statt der hier angebotenen Routine zur Erzeugung eines Mischfarben-Rechteckes können Sie auch das folgende, etwas langsamere Unterprogramm verwenden, das Ihnen auch erlaubt, Rechtecke zu bilden, die nicht unbedingt aus vollständigen 4x8-Punkte-Feldern bestehen:

```
100 PCOL= 1,FS(FA,0) : PCOL= 2,FS(FA,1)
110 FOR Y0=Y1 TO Y2 STEP 2
120   PCOL= 4,1
130   PLOT C ,X1   ,Y0   TO X2,Y0
140   PLOT C ,X1+1,Y0+1 TO X2,Y0+1
150   PCOL= 4,2
160   FOR V=X1 TO X2 STEP 2
170     PLOT C ,V+1,Y0
180     PLOT C ,V  ,Y0+1
190   NEXT V
200 NEXT Y0
210 RETURN
```

Die Übergabeparameter sind identisch, das Ergebnis natürlich auch. Das Prinzip: Eine Linie (bzw. zwei) wird mit Farbregister 1 gezeichnet (Zeilen 120-140) und daraufhin durch einzelne PLOTs mit Farbregister 2 punktiert (Zeilen 150-190).

Auf diese oder ähnliche Art können Sie alle Figuren (Rahmen, Kreise, Punkte etc.) der Supergraphik nachempfinden und sich eine vollständige Graphikerweiterung mit 136 Farben basteln. Die ganz Versierten unter Ihnen nehmen natürlich die entsprechenden Veränderungen an der Supergraphik selbst vor, indem sie das in diesem Buch angegebene Source-Listing der Supergraphik modifizieren.

Ein Beispiel mag das folgende Programm sein, das Vielfarblinien erzeugt:

```
100 REM ***************************
110 REM **                       **
120 REM **  VIEL MEHR FARBEN 2   **
130 REM **                       **
140 REM ***************************
150 REM
160 GMODE 0,2 : GCLEAR
170 SCOL= 0,0
180 SCOL= 1,2 : SCOL= 2,3 : SCOL=3,9
190 REM
200 REM
210 REM FARBKOMBINATIONEN UEBERNEHMEN:
220 REM
230 DIM FS(136,1)
240 FA=0
250 FOR X=0 TO 15      :REM ERSTFARBE
260   FOR Y=X TO 15    :REM ZWEITFARBE
270     READ FS(FA,0),FS(FA,1)
280     FA=FA+1        :REM FARBNUMMER+1
290   NEXT Y
300 NEXT X
310 REM
320 REM
330 REM LINIEN ZEICHNEN:
340 REM
350 F=50
360 FOR X=0 TO 110 STEP 5
370   X1=X : X2=X+45
380   Y1=0 : Y2=190
390   FA=F
400   GOSUB 730      :REM LINIE ZEICHNEN
410   F=F+1          :REM FARBE ERHOEHEN
420 NEXT X
430 REM
440 WAIT 198,255
450 END
460 REM
470 REM
480 REM FELD ZEICHNEN:
490 REM UEBERGABE:
500 REM           FA- FARBE
510 REM           X1- X1-KOORD.
520 REM           Y1- Y1-KOORD.
530 REM           X2- X2-KOORD.
540 REM           Y2- Y2-KOORD.
```

```
550 REM
560 FOR YO=Y1 TO Y2 STEP 2
570   PCOL= 1,FS(FA,0) : PCOL= 2,FS(FA,1)
580   PCOL= 4,3
590   PLOT C B 102,,X1,YO   TO X2,YO
600   PLOT C B 153,,X1,YO+1 TO X2,YO+1
610 NEXT YO
620 RETURN
630 REM
640 REM
650 REM LINIE ZEICHNEN:
660 REM UEBERGABE:
670 REM          FA- FARBE
680 REM          X1- X1-KOORD.
690 REM          Y1- Y1-KOORD.
700 REM          X2- X2-KOORD.
710 REM          Y2- Y2-KOORD.
720 REM
730 PCOL= 1,FS(FA,0) : PCOL= 2,FS(FA,1)
740 PCOL= 4,1
750 PLOT C ,X1,Y1   TO X2,Y2
760 PCOL= 4,2
770 PLOT C ,X1,Y1+1 TO X2,Y2+1
780 RETURN
790 REM
800 REM
810 REM FARBKOMBINATIONSTABELLE:
820 REM
830 DATA  0, 0 ,  0, 1 ,  0, 2 ,  0, 3
840 DATA  0, 4 ,  0, 5 ,  0, 6 ,  0, 7
850 DATA  0, 8 ,  0, 9 ,  0,10 ,  0,11
860 DATA  0,12 ,  0,13 ,  0,14 ,  0,15
870 DATA          1, 1 ,  1, 2 ,  1, 3
880 DATA  1, 4 ,  1, 5 ,  1, 6 ,  1, 7
890 DATA  1, 8 ,  1, 9 ,  1,10 ,  1,11
900 DATA  1,12 ,  1,13 ,  1,14 ,  1,15
910 DATA                  2, 2 ,  2, 3
920 DATA  2, 4 ,  2, 5 ,  2, 6 ,  2, 7
930 DATA  2, 8 ,  2, 9 ,  2,10 ,  2,11
940 DATA  2,12 ,  2,13 ,  2,14 ,  2,15
950 DATA                          3, 3
960 DATA  3, 4 ,  3, 5 ,  3, 6 ,  3, 7
970 DATA  3, 8 ,  3, 9 ,  3,10 ,  3,11
980 DATA  3,12 ,  3,13 ,  3,14 ,  3,15
990 REM
1000 DATA  4, 4 ,  4, 5 ,  4, 6 ,  4, 7
1010 DATA  4, 8 ,  4, 9 ,  4,10 ,  4,11
1020 DATA  4,12 ,  4,13 ,  4,14 ,  4,15
1030 REM
1040 DATA          5, 5 ,  5, 6 ,  5, 7
1050 DATA  5, 8 ,  5, 9 ,  5,10 ,  5,11
```

```
1060 DATA  5,12 ,  5,13 ,  5,14 ,  5,15
1070 REM
1080 DATA                  6, 6 ,  6, 7
1090 DATA  6, 8 ,  6, 9 ,  6,10 ,  6,11
1100 DATA  6,12 ,  6,13 ,  6,14 ,  6,15
1110 REM
1120 DATA                          7, 7
1130 DATA  7, 8 ,  7, 9 ,  7,10 ,  7,11
1140 DATA  7,12 ,  7,13 ,  7,14 ,  7,15
1150 REM
1160 REM
1170 DATA  8, 8 ,  8, 9 ,  8,10 ,  8,11
1180 DATA  8,12 ,  8,13 ,  8,14 ,  8,15
1190 REM
1200 REM
1210 DATA          9, 9 ,  9,10 ,  9,11
1220 DATA  9,12 ,  9,13 ,  9,14 ,  9,15
1230 REM
1240 REM
1250 DATA                 10,10 , 10,11
1260 DATA 10,12 , 10,13 , 10,14 , 10,15
1270 REM
1280 REM
1290 DATA                         11,11
1300 DATA 11,12 , 11,13 , 11,14 , 11,15
1310 REM
1320 REM
1330 REM
1340 DATA 12,12 , 12,13 , 12,14 , 12,15
1350 REM
1360 REM
1370 REM
1380 DATA         13,13 , 13,14 , 13,15
1390 REM
1400 REM
1410 REM
1420 DATA                 14,14 , 14,15
1430 REM
1440 REM
1450 REM
1460 DATA                         15,15
```

In diesem Programm wird ganz einfach eine zweite Linie unter der ersten gezeichnet, was zu dem gewünschten Mischeffekt führt.

2.13 Ein Griff in die Trickkiste - noch mehr Graphikbefehle

2.13.1 Invertieren einmal anders

Ich möchte Ihnen einen neuen Befehl vorstellen:

INVERS

Befehl:	INVERS (m)(,x1,y1 TO x2,y2)
Beispiel:	INVERS INVERS 240 INVERS 240,20,30 TO 60,70
Parameter:	m: Invertierungsmaske x1: x-Koordinate Fenster links oben y1: y-Koordinate Fenster links oben x2: x-Koordinate Fenster rechts unten y2: y-Koordinate Fenster rechts unten
Funktion:	Invertieren des Graphikbildes oder eines Graphikfensters mit 256 verschieden Masken

---

Es handelt sich dabei um einen Befehl, der Ihre HGR- oder MC-Graphiken invertiert. Das Invertieren kennen Sie vielleicht bereits von den normalen Zeichenbefehlen. Dort konnten Sie mit zm=2 den Invertier-Modus einschalten, d.h. im HGR-Modus wird jeder nicht gesetzte Punkt gesetzt und umgekehrt, in MC dagegen findet ein Farbnummernwechsel statt. Der Befehl INVERS invertiert nun ein ganzes Graphikbild oder, falls die Fensterkoordinaten angegeben wurden, ein Graphikfenster. Dabei enthält er jedoch eine ganz besondere Raffinesse:

Mit m (eine Zahl zwischen 0 und 255) geben Sie an (interessant ist die dem Wert entsprechende Dualzahl, s. auch GCLEAR), welche Bits eines Bytes des Graphikspeichers (8 nebeneinander liegende Punkte) jeweils invertiert werden sollen. So kann die gesamte Graphikseite z.B. gestreift invertiert werden.
Sie gehen also folgendermaßen vor: Denken Sie sich eine 8-Punkte-Maske, mit der invertiert werden soll. Jeder gesetzte Punkt der Maske verursacht nachher beim Invertieren ein Umkehren des jeweiligen Punktes auf dem Bildschirm, ein nicht gesetzter Punkt hat keinen Einfluß auf das Bild. Unter Multicolor sieht das Ganze ein wenig komplizierter aus, denn hier müssen Sie jeweils 2 Punkte dieser Maske zusammenfassen, da sie in der Graphik später auf einen Punkt treffen.

Nehmen wir die Maske:

.xx.xx..

Wir übersetzen diese Punktmaske zunächst in eine Binärzahl:

01101100

und dann in eine dezimale Zahl:

%01101100 = 108

Diese dezimale Zahl ist dann unsere Maske m.

Durch zweimaliges gleiches Invertieren wird der Ursprungszustand wieder hergestellt. Invertieren mit dem Wert 255 entspricht damit einer vollständigen Invertierung. Am besten Sie probieren alles einmal aus:

```
100 REM ****************
110 REM **            **
120 REM **  INVERSER  **
130 REM **            **
140 REM ****************
150 REM
160 GMODE 0,1 : GCLEAR
170 COLOR= 0,0
180 SCOL= 1,5
190 TEXT ,"SUPERGRAPHIK 64",80,10,0
200 FOR X=0 TO 198 STEP 5
210   PLOT ,X,X TO 50,X
220   PLOT ,X,X+1 TO 51,X+1
230 NEXT X
240 REM
250 FOR M=1 TO 6
260   INVERS M-1
270   INVERS M
280 NEXT M
290 INVERS M-1
300 REM
310 INVERS 160
320 FOR X=1 TO 5000 : NEXT X
330 INVERS 160
340 REM
350 REM
360 REM *****************
```

```
370 REM **                **
380 REM **  MULTICOLOR  **
390 REM **         `      **
400 REM ******************
410 REM
420 GMODE ,2 : GCLEAR
430 SCOL= 0,0
440 SCOL= 1,8 : SCOL= 2,5 : SCOL=3,6
450 PCOL= 4,1
460 FRAME ,10,10,10 TO 60,60
470 PCOL= 4,2
480 FRAME ,10,40,40 TO 90,90
490 PCOL= 4,3
500 FRAME ,10,80,80 TO 120,120
510 REM
520 FOR X=1 TO 2000 : NEXT X
530 INVERS 240
540 FOR X=1 TO 5000 : NEXT X
550 REM
560 REM
570 REM ***************************
580 REM **                       **
590 REM **  FENSTER INVERTIEREN  **
600 REM **                       **
610 REM ***************************
620 REM
630 GCLEAR 130, 30, 50 TO 220,180
640 INVERS  30, 60, 70 TO 200,170
650 REM
660 WAIT 198,255
```

Das obige Programm sollte nicht allzu schwer zu verstehen sein. Bemerkenswert ist der zweite Teil der kleinen Demo ab Zeile 420. Hier werden Sie Zeuge eines Farbwechsels in Multicolor durch die Invertierung. Dieser neue Befehl wird die Möglichkeiten der Supergraphik sicher weiter steigern.

Ebenso wie beim GCLEAR-Befehl können Sie beim Befehl INVERS die Eckkoordinaten eines beliebigen Graphikfensters angeben. In diesem Fall wird nur das angegebene Bildschirmfenster invertiert. Wie Sie Bildschirmfenster anwenden, erfahren Sie ein Kapitel weiter beim GCOMB-Befehl.

2.13.2 Kopieren und Überlagern - Aus zwei mach eins und eins ist keins

Bildschirmfenster

Oft verlangen spezielle Aufgaben, wie z.B. die Vernetzung dreidimensionaler Funktionen, Unterschiedserkennungen oder einfach Bilderaddition, die Verknüpfung zweier Graphikbilder durch ODER, UND o.ä. Funktionen.

In letzter Zeit werden aber auch durch die Einführung leistungfähiger 16-Bit-Rechner immer höhere Ansprüche an die Benutzerfreundlichkeit gestellt. Gefordert wird die Handhabung von sogenannten Bildschirmfenstern. Auch hier kann die Supergraphik umfangreiche Unterstützung gewähren. Zwei Befehle - INVERS und GCLEAR - haben Sie bereits kennengelernt, die die Möglichkeit schaffen, mit Bildschirmfenstern zu hantieren. Wir haben diese Fähigkeit der beiden Befehle bei der jeweiligen Vorstellung bisher nur kurz erwähnt und völlig ungebührend vernachlässigt. Das soll hier nun nachgeholt werden im Zusammenhang mit einem weiteren Befehl dieser Gruppe, der das Bildschirmfensterhandling vervollständigt:

GCOMB

Befehl:	GCOMB m (,x1,y1 TO x2,y2 (TO x3,y3))	
Beispiel:	GCOMB 1	
	GCOMB 2,20,30 TO 50,120	
	GCOMB 3,40,10 TO 90,130 TO 30,50	
Parameter:	m:	Verknüpfungsmodus (0-7)
	x1:	x-Koordinate Fenster links oben
	y1:	y-Koordinate Fenster links oben
	x2:	x-Koordinate Fenster rechts unten
	2:	y-Koordinate Fenster rechts unten
	3:	x-Koordinate Zielfenster links oben
	3:	y-Koordinate Zielfenster links oben
Funktion:	Verknüpfung beider Graphikseiten, zweier Bildschirmfenster, Verschiebung eines Graphikfensters	

Der GCOMB-Befehl ermöglicht Ihnen die Bewältigung dieser Aufgabe. Er verknüpft jeweils zwei Graphikseiten (oder Teile davon) und speichert das Ergebnis in der aktuellen Seite. Wie Sie die beiden Graphikseiten ansprechen, haben Sie bereits im Kapitel 2.5.4. kennengelernt. Für m sind Werte zwischen 0 und 7 zulässig, die jeweils unterschiedliche Verknüpfungsarten bestimmen:

```
m=0, m=4: UND-Verknüpfung zweier Seiten
```

in HGR:

Nur dort wird ein Punkt gesetzt, wo in beiden Graphikseiten ein Punkt steht.

in MC:

Ist in einer der beiden Seiten ein Punkt in Hintergrundfarbe gesetzt oder ist in der einen Seite ein Punkt in Farbregisternummer 1, in der anderen in Registernummer 2 gezeichnet, so resultiert an dieser Stelle im Ergebnisbild ein Punkt in Hintergrundfarbe.

Ist in beiden Graphikseiten an einer Stelle ein Punkt in derselben Farbregisternummer gesetzt, so nimmt der resultierende Punkt an dieser Stelle ebenfalls diese Farbregisternummer an. Ist in einer Seite ein Punkt in Farbe 3 gesetzt, so erscheint im Ergebnisbild ein Punkt in der Farbe des Punktes, der an dieser Stelle in der anderen Seite stand.

m=1, m=5: ODER-Verknüpfung zweier Seiten

in HGR:

Überall dort, wo in Seite 1 und/oder in Seite 2 ein Punkt steht, wird ein Punkt gesetzt.

in MC:

Ist in einer der beiden Seiten ein Punkt in Farbe 3 gesetzt, oder ist in der einen Seite ein Punkt in Farbregisternummer 1, in der anderen in Registernummer 2 gezeichnet, so resultiert an dieser Stelle im Ergebnisbild ein Punkt in Farbe 3. Ist in beiden Graphikseiten an einer Stelle ein Punkt in derselben Farbregisternummer gesetzt, so nimmt der resultierende Punkt an dieser Stelle ebenfalls diese Farbnummer an. Ist in einer Seite ein Punkt in Hintergrundfarbe gesetzt, so erscheint im Ergebnisbild ein Punkt in der Farbe des Punktes, der an dieser Stelle in der anderen Seite stand.

m=2, m=6: EXKLUSIV-ODER-Verknüpfung der Seiten

in HGR:

Nur dort, wo lediglich in einer Seite ein Punkt gesetzt ist, wird auch im Ergebnisbild ein Punkt gesetzt.

in MC:

Ist in einer der beiden Seiten ein Punkt in Farbe 3 gesetzt oder sind an dieser Stelle beide Punkte in derselben Farbe gezeichnet, so erscheint im Ergebnisbild ein Punkt in der inversen Farbe des Punktes der anderen Seite (s. INVERS). Steht in der einen Seite ein Punkt mit Registernummer 1, in der anderen mit Registernummer 2, so erscheint im Ergebnisspeicher dort ein Punkt in Farbregisternummer 3. Ist

dagegen in einer der beiden Seiten ein Punkt in der Hintergrundfarbe gesetzt, so erscheint im Ergebnisbild ein Punkt mit der gleichen Farbregisternummer des Punktes der anderen Seite.

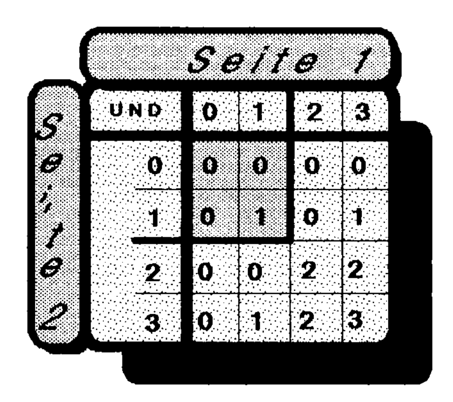
```
m=3, m=7: Kopieren eines Fensters in die andere Seite
```

in HGR und MC:

Ein Bildschirmfenster oder eine ganze Seite wird vollständig und identisch in die befehligte Seite übertragen.

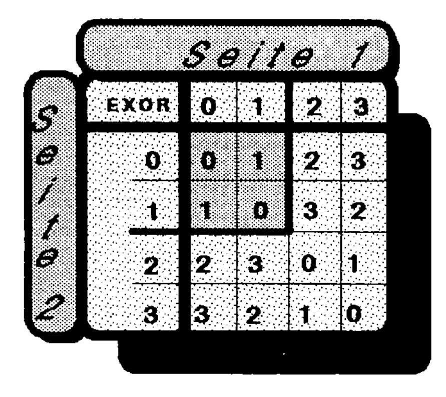
Diese vielleicht etwas komplizierten Sachverhalte wollen wir in den folgenden 3 Tabellen noch einmal verdeutlichen:





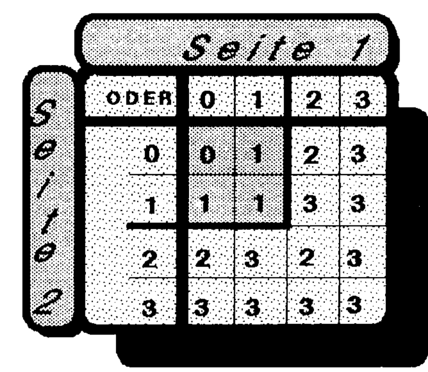
Seite 1 / Seite 2

UND	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	2	2
3	0	1	2	3



Seite 1 / Seite 2

EXOR	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0



Seite 1 / Seite 2

ODER	0	1	2	3
0	0	1	2	3
1	1	1	3	3
2	2	3	2	3
3	3	3	3	3

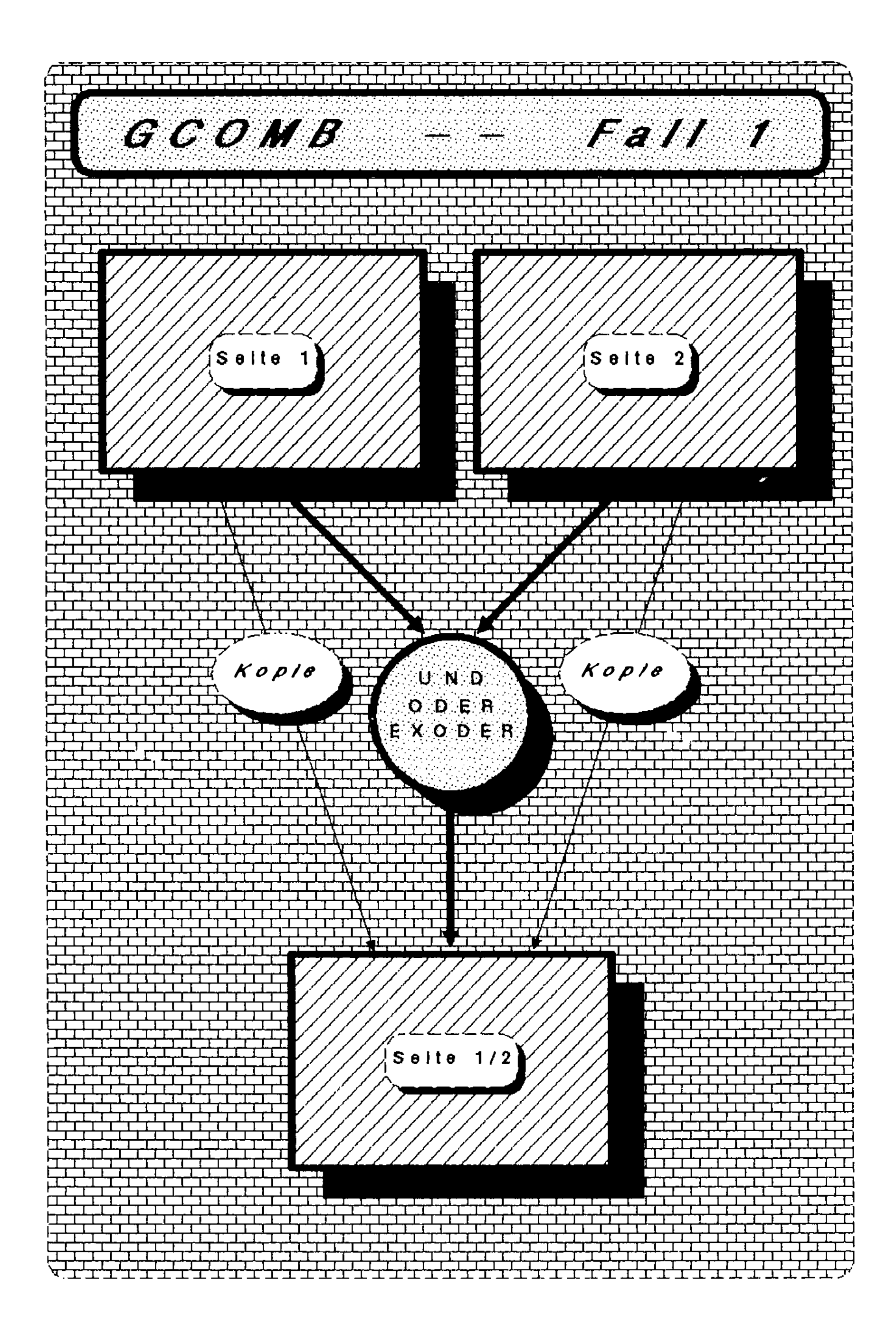
Die vollständigen Tabellen sind jeweils für die Fälle m=0,1,2,4,5,6 in Multicolor. Die kleinen fett umrandeten Vierer-Quadrate sind jeweils für HGR. Die Spalten ganz rechts und die Reihen ganz oben beinhalten jeweils die Registernummern eines Punktes in der einen und in der anderen Graphikseite. Die Schnittpunkte der Spalten und Reihen geben dann jeweils die Registernummer an, die der jeweilige Punkt im Ergebnisbild erhält.

Warum aber zwei Modusnummern für eine Verknüpfungsart? Nun, das ganze hängt mit der ebenfalls notwendigen Farbgebung zusammen. Mit der niedrigeren der beiden Nummern geben Sie an, daß die Farbe aus der Graphikseite stammen soll, in die das Ergebnisbild gespeichert wird. Geben Sie dagegen die höhere Nummer an, so wird die Farbe aus der Quellseite mit in die Zielseite übertragen. Dabei ist jedoch zu beachten, daß die Farbe sich jeweils auf vollständige 4x8- bzw. 8x8-Felder bezieht. Liegt also die Grenze eines Bildschirmfensters irgendwo innerhalb eines solchen Feldes, so wird die Farbe der Punkte außerhalb des Fensters in der Nähe des Randes mit beeinflußt.

Kommen wir nun zur eigentlichen Fensterbehandlung. Wie gesagt, geben Sie (ähnlich dem FILL- oder FRAME-Befehl mit den Koordinaten x1,y1 die linke obere und mit x2,y2 die rechte untere Ecke des angesprochenen Graphikfensters an. Bei den Befehlen GCLEAR und INVERS wird dieses Graphikfenster einfach gelöscht oder invertiert. Mit dem GCOMB-Befehl verhält es sich doch ein wenig komplizierter:

1. Fall:

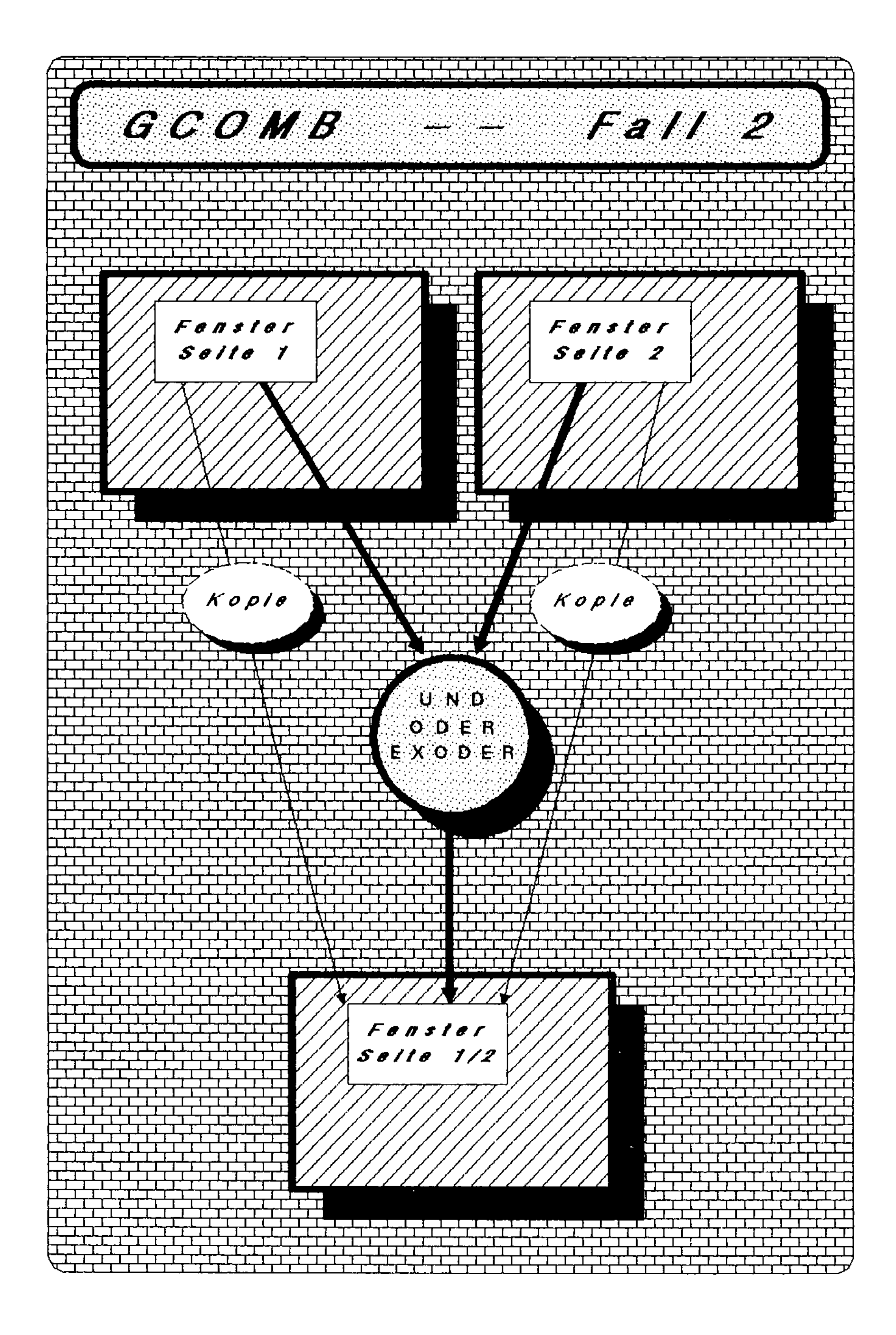
Im ersten Fall lassen Sie die Fensterkoordinaten weg. Es werden also stets die beiden vollen Graphikseiten miteinander verknüpft. Das Ergebnis steht dann in der befehligten Graphikseite.


GCOMB -- Fall 1
Seite 1
Seite 2
Kopie
UND
ODER
EXODER
Kopie
Seite 1/2

2. Fall:

Angenommen Sie geben tatsächlich nur die beiden Koordinaten x1,y1 und x2,y2 an. In diesem Fall bezieht sich die Position und Größe des Bildschirmfensters auf beide Graphikseiten, d.h. sowohl das Fenster der Seite 1, als auch das auf Seite 2 stehen an der selben Stelle auf dem Bildschirm. Die beiden Fenster werden nun auf die angegebene Weise verknüpft und das Ergebnis steht in dem Fenster derjenigen Graphikseite, die gerade befehligt wird.

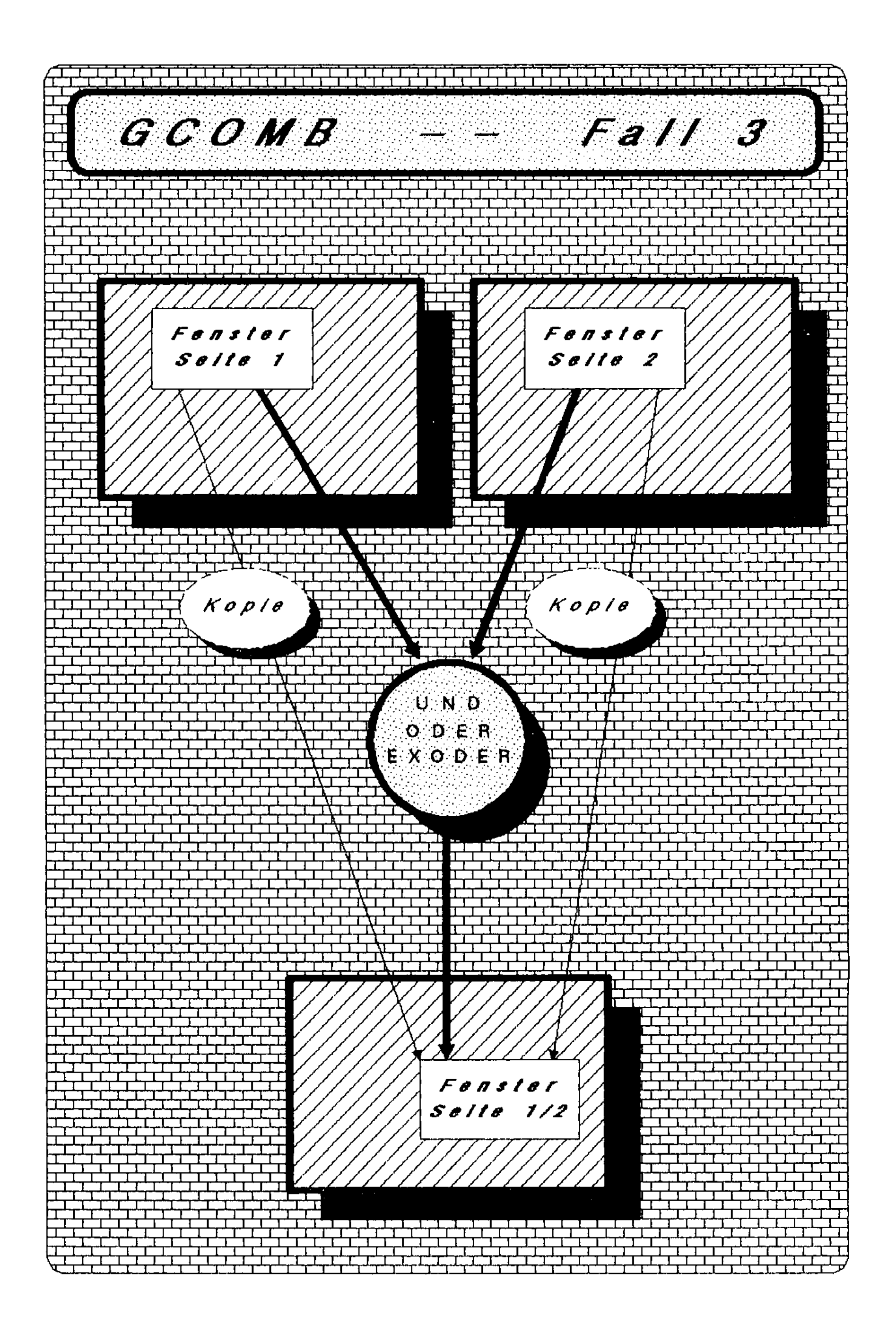
Interessant ist auch Modus 3 bzw. 7. Hierbei wird ein Graphikfenster vollständig in eine andere Seite kopiert. Hier kann es z.B. zwischengespeichert werden, um später wieder in die Ursprungsseite zurückkopiert zu werden. Diese Technik ist wesentlich für den Umgang mit Bildschirmfenstern. Doch das werden wir noch sehen.


GCOMB -- Fall 2
Fenster Seite 1
Fenster Seite 2
Kopie
Kopie
UND
ODER
EXODER
Fenster Seite 1/2

3. Fall:

Sie geben zusätzlich zu den beiden Koordinaten x1,y1 und x2,y2 noch die Zielkoordinaten x3,y3 an. Diese Zielkoordinaten bestimmen ein zweites, gleichgroßes Graphikfenster mit der oberen linken Ecke in x3,y3 an. Die anderen Ecken werden logischerweise automatisch durch die Größe des ersten Fensters bestimmt. Beim GCOMB-Befehl wird nun das erste Fenster (x1,y1/x2,y2), das in der nicht befehligten Seite liegt, mit dem zweiten Fenster (x3,y3), das in der befehligten, also der Zielseite liegt, auf die bekannte Weise kopiert.

Auch hier ist Modus 3 (7) wieder interessant für die Fenstertechnik. Mithilfe dieses Modus können Sie nämlich ohne Probleme Bereiche des Bildschirms verschieben, indem Sie ein Fenster erst in die eine Seite kopieren und dann verschoben in die erste zurückkopieren.


GCOMB -- Fall 3
Fenster
Seite 1
Fenster
Seite 2
Kopie
Kopie
UND
ODER
EXODER
Fenster
Seite 1/2

Bleiben noch die verschiedenen Syntaxmöglichkeiten zu erwähnen:

```
GCOMB m
GCOMB m,x1,y1 TO x2,y2
GCOMB m,x1,y1 TO x2,y2 TO x3,y3
```

Im Kapitel 3.4 erhalten Sie einen umfassenden Überblick über die programmiertechnische Realisierung der Fensterbefehle.

2.13.3 Scrollen und Rollen - wir rotieren

Bevor wir uns den Ausgabebefehlen zuwenden, möchte ich Ihnen noch einen letzten direkten Graphikbefehl vorstellen, der ebenfalls ein klein wenig Außergewöhnlichkeit besitzt:

GMOVE

Befehl:	GMOVE m,za,ze	
Beispiel:	GMOVE 0,10,20	
Parameter:	m:	Verschiebemodus (0-3)
	za:	Anfangszeile (0-24)
	ze:	Endzeile (0-24)
Funktion:	Horizontales Rollen/Verschieben des Bildschirms	

Dieser Befehl, so klein er ist, ist eine weitere Spezialität der Supergraphik 64. Er ermöglicht Ihnen, Teile des Bildschirms (LGR, HGR oder MC) schnell und einfach nach rechts oder links zu verschieben; dabei gibt m die Richtung und die Scroll-Art an:

m=0: Links rollen
m=1: Rechts rollen
m=2: Links verschieben
m=3: Rechts verschieben

Beim Rollen des Bildschirms (oder von Teilen des Bildschirms) erscheint alles, was über den linken oder rechten Rand hinaus geht, auf der anderen Seite wieder. Das Verschieben dagegen zeichnet sich dadurch aus, daß diese Teile unwiederbringbar verschwinden.

Mit dem 2. Wert geben Sie die Nummer der Anfangszeile an, mit dem 3. die Endzeile von denen aus gescrollt werden soll.
Mit Zeile ist im HGR- und MC-Modus ein 8-Punkte dicker Streifen gemeint, der einer normalen Textzeile entspricht. Im LGR-Modus besteht somit eine Zeile aus zwei-Punktestreifen bzw. einer normalen Textzeile. Man unterteilt den Bildschirm somit einfach wie im Text in 25 Zeilen.

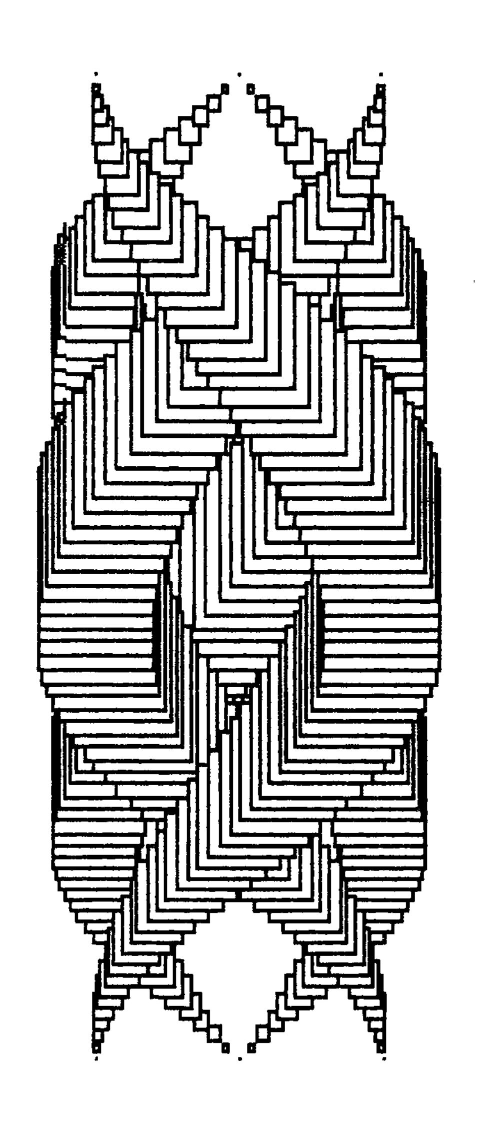
Welche Möglichkeiten durch diesen Befehl entstehen, sei hier nur von Programmierermund zu Programmiererohr geflüstert:
Aktionspiele mit beweglichem Hintergrund, Laufschriften und vieles mehr. Besonders im Text-Modus findet dieses Rollen mit enormer Geschwindigkeit statt und erlaubt unglaubliche Effekte.

```
100 REM ******************************
110 REM **                          **
120 REM **  ROLLEN, SCROLLEN, ...   **
130 REM **                          **
140 REM ******************************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 1,0 : COLOR= 6,6
170 PI=3.14159265/180
180 DEF FN F(I)=200+100*SIN(PI*I)
190 GOSUB430
200 DEF FN F(I)=200-100*SIN(PI*I)
210 GOSUB430
220 DEF FN F(I)=200+100*COS(PI*I)
230 GOSUB430
240 DEF FN F(I)=200-100*COS(PI*I)
250 GOSUB430
260 REM
270 FOR X=1 TO 2000 : NEXT X
280 REM
290 REM ROLLEN/SCROLLEN:
300 REM
310 FOR X=1 TO 40
320   GMOVE 1,3,21
330 NEXT X
340 FOR X=1 TO 40
350   GMOVE 3,3,21
360 NEXT X
380 END
390 REM
400 REM
410 REM ZEICHENROUTINE:
420 REM
430 FOR I=0 TO 360 STEP 5
440   X=50+I*1.5
```

```
450   Y=FN F(I)
460   DD=40*SIN(PI*I/2)
470   X1=(X+20+DD)/2 : X2=(X+20-DD)/2
480   Y1=(Y+DD)/2    : Y2=(Y-DD)/2
490   FRAME  ,1,X1,Y1 TO X2,Y2
500   FRAME 1,4,X1-1,Y1-1 TO X2+1,Y2+1
510 NEXT I
520 RETURN
```

Versuchen Sie doch einmal dieses Programm:

```
10 GMODE 0,1 : GCLEAR
20 TEXT ,"SUPERGRAPHIK 64",100,7,0
30 FOR X=1 TO 400
40   GMOVE 0,0,0
50   FOR Y=X TO 150 : NEXT Y
60 NEXT X
70 WAIT 198,255
```



oder:

```
100 REM *****************
110 REM **             **
120 REM **  ROLL-DEMO  **
130 REM **             **
140 REM *****************
150 REM
160 COLOR= 0,0
170 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
180 POS= 12,1
190 PRINT "SUPERGRAPHIK 64"
200 GOSUB 530
210 POS= 14,5
220 PRINT "MACHT ALLES"
230 GOSUB 530
240 POS= 17,7
250 PRINT "VIEL"
260 GOSUB 530
270 POS=  3,11
280 PRINT "SCHNELLER, SCHNELLER, SCHNELLER ..."
290 REM
300 REM
310 REM ROTIEREN:
320 REM *********
330 REM
340 FOR X=1 TO 520
350   GMOVE 0,11,11
360   FOR T=X TO 190 : NEXT T
370 NEXT X
380 REM
390 GOSUB 530
400 POS= 15,14
410 PRINT "ALLES!!!"
420 GOSUB 530
430 REM
440 FOR X=1 TO 400
450   GMOVE 1,1,14
460 NEXT X
470 END
480 REM
490 REM
500 REM WARTESCHLEIFE:
510 REM **************
520 REM
530 FOR T=1 TO 500 : NEXT T
540 RETURN
```

Na, überzeugt?

2.14 Wir geben aus

Bisher haben wir uns stets mit Befehlen beschäftigt, die die schönsten Graphiken auf den Bildschirm bringen. Sie können nun schon Punkte, Linien, Kreise, Ellipsen, Rahmen, Text usw. zeichnen, löschen, invertieren und sonst noch was. Was nützen Ihnen aber die schönsten Graphiken, die vielleicht in stundenlanger Programm-Arbeit auf Ihrem Computer entstanden sind, wenn alles wieder verloren ist, indem Sie den Rechner ausschalten?

Wir werden im folgenden eine Reihe von sehr nützlichen Befehlen kennenlernen, die es Ihnen ermöglichen, Graphiken auf Diskette zu speichern oder wieder zu laden, Graphiken fremder Graphikprogramme mit der Supergraphik weiterzubearbeiten, Graphiken auf dem Drucker auszugeben usw. Sie haben damit die Möglichkeit, sozusagen mit der Außenwelt in Verbindung zu treten.

2.14.1 Laden und Speichern von Graphiken

Zunächst wollen wir uns mit der Konservierung und Archivierung unserer Graphiken auf Diskette oder Datasette beschäftigen. Bei den folgenden Befehlen wurde wieder die Technik der Befehlsmodiwahl angewandt, um möglichst viele Funktionen überschaubar anzubieten. Dabei entsprechen sich die Befehle GSAVE und GLOAD in dieser Beziehung, sie besitzen die gleiche Befehlssyntax:

GSAVE

Befehl: GSAVE (LGCF),s,"filename"(,ga)
Beispiel: GSAVE GCF,1,"MULTICOLOR",8
Parameter: LGCF: Formatsteuerung (s.u.)
s: Angesteuerte Graphikseite
ga: Geräteadresse
Funktion: Speichern der Graphik, Farbe oder Text

Dieser Befehl speichert die mit n (1 oder 2) angegebene Graphikseite (unabhängig davon, welche angezeigt oder befehligt wird) unter dem ebenfalls angegebenen Filenamen auf Kassette (ga=1; kann auch weggelassen werden) oder Diskette (ga=8). Damit haben Sie die Möglichkeit, einmal erstellte Graphiken zu sichern.

Sie können sich das Format Ihrer Graphikfiles selbst festlegen, d.h. Sie können wählen, welche Graphik- oder Farbteile in einem gemeinsamen File und in welcher Reihenfolge abgespeichert werden. Damit erreichen Sie eine maximale Kompatibilität zu Bildern anderer Graphikprogramme. Es gibt die Möglichkeit 4 verschiedene Speicherbereiche auf Diskette zu bringen. Jedem Bereich ist ein Buchstabe zugeordnet, der in der Formatanweisung erscheinen muß:

L:	Low-Graphik (Text)	(1000 Bytes)
G:	Graphikspeicher	(8000 Bytes)
C:	Farbvideo-RAM	(1000 Bytes)
F:	Farb-RAM	(1000 Bytes)

Diese Buchstaben können nun beliebig kombiniert oder ausgelassen werden, wobei ihre Reihenfolge die Reihenfolge der verschiedenen Bereiche in dem entstehenden File beschreibt. Hier einige gebräuchliche Kombinationen:

GSAVE LF,1,"TEXT",8
Speichern des Textes mit Farbe auf Diskette.

GSAVE GC,1,"HGR MIT FARBE",8
Speichern der hochauflösenden Graphik Seite 1 mit Farbe.

GSAVE G,2,"HGR OHNE FARBE",8
Speichern lediglich des Graphikspeichers ohne Farbe (Seite 2).

GSAVE GCF,1,"KOALA-PAD",8
Speichern eines Multicolor-Bildes der Graphikseite 1 mit Farbe im KOALA-PAD-Format.

Natürlich können alle anderen Kombinationen gewählt werden. Dabei dürfen allerdings maximal 4 Buchstaben hintereinander verwendet werden.

Die gleiche Syntax besitzt der Befehl zum Laden von Graphik von Diskette:

GLOAD

Befehl:	GLOAD (LGCF),s,"filename"(,ga)
Beispiel:	GLOAD GCF,1,"MULTICOLOR",8
Parameter:	LGCF: Formatsteuerung (s.u.)
	s: Angesteuerte Graphikseite
	ga: Geräteadresse
Funktion:	Laden der Graphik, Farbe oder Text

Wollen Sie eine einmal abgespeicherte Graphik wieder laden, so verwenden Sie GLOAD. Die Parameter haben dieselbe Bedeutung wie bei GSAVE. Nun können Sie beispielsweise verdeckt eine Graphik in die unsichtbare Seite einladen, um dann direkt auf diese Seite umzuschalten. So lassen sich z.B. ganze Bilderreihen nacheinander schnell quasi als Trickfilm oder Dia-show einladen und zeigen.

Es empfiehlt sich, eine in einem bestimmten Format abgespeicherte Graphik in demselben Format auch wieder einzuladen, obwohl das in manchen Fällen nicht notwendig ist (nur die Reihenfolge der Buchstaben muß stimmen).

Das folgende Programm zeichnet eine kleine Graphik auf den Bildschirm, speichert sie auf Diskette (bitte Diskette ins Laufwerk legen), löscht den Graphikbildschirm und läd die Graphik wieder von Diskette in den Rechner. Um das Programm für die Datasette lauffähig zu machen, müssen Sie lediglich die Geräteadresse ga, also die letzte Zahl des Lade- oder Speicherbefehls von 8 in 7 umwandeln:

```
10 GMODE 0,1 : GCLEAR
20 FILL 3,40,50 TO 60,70
30 FILL 2,10,60 TO 80,90  :REM DEMO ZEICHNEN
40 GSAVE G,1,"TESTBILD",8 :REM BILD AUF DISKETTE SPEICHERN
50 GCLEAR                 :REM GRAPHIK LOESCHEN
60 GLOAD G,1,"TESTBILD",8 :REM GRAPHIK WIEDER LADEN
70 WAIT 198,255
```

2.14.2 Einladen von Fremdgraphiken

Die große Flexibilität der beiden Befehle GLOAD und GSAVE erlaubt es, eine ganze Reihe von Graphiken in die Supergraphik einzuladen, die nicht mit der Supergraphik erstellt wurden. Im folgenden wollen wir kurz zeigen, wie Sie mit der Supergraphik Graphiken aus zwei verschiedenen Graphikprogrammen einladen oder für diese Programme abspeichern können. Bei anderen Programmen müssen Sie ein wenig probieren, um das richtige Format herauszubekommen:

a) Koala-Pad:

Eines der ältesten und bekanntesten Graphikprogramme ist das Menue-gesteuerte Programm zum Koala-Pad. Unter Koala werden grundsätzlich nur Multicolorbilder bearbeitet. Aus diesem Grunde können wir auch nur Bilder in Multicolor abspeichern oder laden. Da im normalen Text-Modus die Textfarbe im Farb-RAM beherbergt wird, der aber unter MC für die Farbe 3 zuständig ist, empfiehlt es sich, Koala- und allgemein Multicolorbilder nicht einfach im Eingabemodus, sondern per Programm abzuspeichern oder zu laden. Hier ein Programm, das ein Koala-Pad-Bild in die Supergraphik einläd:

```
10 GMODE 0,2 : GCLEAR :REM MULTICOLOR EIN UND LOESCHEN
20 GLOAD GCF,1,CHR$(129)+"PIC A BILD",8 :REM EINLADEN
30 SCOL= 0,1          :REM HINTERGRUNDFARBE BESTIMMEN
40 WAIT 198,255
```

Die Buchstabenkombination GCF sorgt dafür, daß nacheinander Graphikspeicher, Video-RAM und Farb-RAM eingeladen werden, wie es bei Koala-Bildern notwendig ist. Lediglich die Hintergrundfarbe müssen Sie, wie hier in Zeile 30, selbst anwählen (in diesem Fall weiß).

Der Ausdruck CHR$(129)+ ist notwendig, da Koala-Pad-Bilder dieses Zeichen zu Anfang jedes Bildnamens als Kennzeichen besitzen.

Wollen Sie ein Bild abspeichern, so verwenden Sie das gleiche Programm und ersetzten den GLOAD-Befehl durch GSAVE.
Die Ausführungen gelten natürlich auch für andere Programme, die Multicolorbilder erzeugen.

b) Hi-Eddi plus:

Hi-Eddi plus ist ein neueres Graphikprogramm, mit dem Sie ebenfalls Graphiken erzeugen, laden und abspeichern können. Es ist keine Befehlserweiterung wie die Supergraphik, sondern ein durch Tastenkürzel gesteuertes Programm. Mit diesem Programm können Sie zwei Arten von Bildern erzeugen:

- Graphikbild
- Farbbild

Beim Graphikbild wird nur die reine Graphik ohne Farbe abgespeichert. Sie können Sie durch ein:

```
GMODE 0,1 : GLOAD G,1,"BILD.PIC",8
```

einladen. Da für die Graphik bei diesem Programm statt der notwendigen 8000 Bytes einfachheitshalber 8128 Bytes abgespeichert werden, ist ein Einladen des Farbbildes nicht ohne weiteres möglich. Das folgende Programm löst aber auch dieses Problem. Es dauert zwar etwas länger, aber Sie haben damit zumindest die Möglichkeit, ein Farbbild einzuladen:

```
100 GMODE 0,1 : GCLEAR
110 N$="BILD.PIC"          :REM BILDNAME
120 OPEN 1,8,2,N$          :REM DATEI OEFFNEN
130 GET#1,A$ : GET#1,A$    :REM 2 BYTES UEBERSCHLAGEN
140 REM
150 REM GRAPHIK EINLESEN:
160 REM
170 FOR X=14*4096 TO 14*4096+8217
180   GET#1,A$ : POKE X,ASC(A$+CHR$(0))
190 NEXT X
200 REM
210 REM FARBE EINLESEN:
220 REM
230 FOR X=12*4096 TO 12*4096+1000
240   GET#1,A$ : POKE X,ASC(A$+CHR$(0))
250 NEXT X
260 REM
270 CLOSE 1 :REM DATEI SCHLIESSEN
```

Die Ausführungen zu Hi-Eddi-Plus (samt Programm etc.) gelten im übrigen für alle Programme, die ihre Graphik auf diese Weise abspeichern. Kennzeichen dafür sind reine Graphikfiles auf Diskette mit 33 Blöcken, bei HGR-Graphiken mit Farbe entstünden dann auf Diskette Files mit 37 Blöcken (bei MC-Files 41 Blöcke). Programme, die nur die notwendigen 8000 Bytes abspeichern (wie die Supergraphik), erzeugen auf Diskette 32 Blöcke-Files für reine Graphik oder 36 Blöcke für Farbbilder (bei MC-Files 40 Blöcke).

2.14.3 Ausgabe von Graphik auf dem Drucker

Krönung jeder Graphik ist ihre Ausgabe auf einem graphikfähigen Drucker. So hat auch die Supergraphik die Möglichkeit, Graphiken auf einem Matrixdrucker auszugeben. Das leidige und alte Problem aller Graphikprogramme in diesem Zusammenhang ist die Vielzahl der Drucker und speziell beim C64 die unterschiedlichen Interfaces für die verschiedensten Drucker. So wird es kaum möglich sein, Hardcopy-Routinen, Routinen also, die den Bildschirm auf Drucker ausgeben, für jeden denkbaren Drucker zu erstellen.

Trotzdem oder gerade deshalb hat sich der Autor der Supergraphik bemüht, möglichst viele Drucker mit der Graphikausgabe harmonisieren zu lassen. Hierzu wurden bisher 3 verschiedene Hardcopyroutinen implementiert, die Hardcopies auf den folgenden Druckern zulassen:

- alle Epson-kompatiblen mit DATA-BECKER-Interface
- MPS 801
- MPS 803
- CBM 1525
- Seikosha GP 100VC
- MPS 802
- CBM 1526
- Farbdrucker Seikosha GP 700

Im letzten Fall wird sogar eine vollständige Farb-Hardcopy auf dem Farbdrucker ausgegeben.

Die jeweiligen Hardcopy-Routinen besitzen in der Supergraphik einen festen Speicherplatz und werden je nach Drucker nachgeladen (s. Startmenue der Supergraphik). Hierdurch können sich - wenn Sie ein paar Regeln einhalten - selbst eine Hardcopy-Routine für Ihren eigenen Drucker schreiben, auf Diskette ablegen und durch das Startmenue aufrufen lassen. So finden Sie z.B. im Graphikbuch zum

C128, das auch für 64er-Freunde interessant sein dürfte, eine Hardcopy speziell für alle Epson-Drucker, die ganz speziell auf die Möglichkeiten dieser Drucker eingeht. Eine ähnliche, nicht ganz so komfortable Epson-Hardcopy finden Sie aber auch im 64 intern. Wenn Sie tatsächlich eine eigene Hardcopy-Routine einbinden wollen, so schauen Sie sich hierzu die Source-Listings der bestehenden Routinen der Supergraphik in diesem Buch an. Dort erfahren Sie, wie Sie vorzugehen haben.

Doch nun sollten wir uns dem Befehl hingeben, der diese Freuden und Prohpezeihungen realisiert:

HCOPY

Befehl:	HCOPY# n
Beispiel:	HCOPY# 1
Parameter:	n: logische Filenummer
Funktion:	Fertigen einer Hardcopy der Graphikanzeige

Falls Sie keinen Drucker besitzen, können Sie diesen Befehl natürlich überschlagen, sollten sich aber unbedingt baldigst einen zulegen. Sie werden es nicht bereuen! Eine schön ausgedruckte Graphik ist - wie gesagt - der Gipfel und letztendlich das Ziel jeder Graphikprogrammierung.

Die verschiedenen Hardcopy-Routinen für die diversen Drucker besitzen teilweise unterschiedliche Syntax. So ist die oben angegebene Schreibweise des Befehls mit dem Parameter n für alle Drucker außer dem Farbdrucker gültig.

Im folgenden seien die zur Zeit vorhandenen Routinen erläutert.

Vor jedem Hardcopy-Befehl muß der Drucker betriebsbereit gemacht und ein entsprechender Kanal mit OPEN geöffnet werden. Z.B.:

```
10 OPEN 1,4
20 HCOPY#1
30 CLOSE 1
```

Auch der OPEN-Befehl kann von Drucker zu Drucker variieren. Sehen Sie dazu in Ihrem Drucker- bzw. Interface-Handbuch nach. So gilt das obige Programm für die Drucker:

MPS 801-803, CMB 1525/1526, Seikosha GP 100VC

Wenn Sie das richtige Interface besitzen (z.B. DATA BECKER), so gilt es auch für Epson-Drucker.

Farbdrucker Seikosha GP 700:

Von den drei Hardcopy-Routinen ragt eine in ganz besonderem Maße hervor: Die farbige Hardcopy auf dem Seikosha GP-700.

Zur Farbgebung ist folgendes zu beachten: Es wurde der sogenannte "head-hammer"-Farbmodus ausgewählt, d.h. es stehen Ihnen folgende 8 Farben zur Verfügung:

0 = weiß (kein Druck)
1 = gelb
2 = purpurrot
3 = rot-orange
4 = hell-blau
5 = grün
6 = purpur (violett)
7 = schwarz

Um Ihnen die Arbeit zu erleichtern, ordnet die Supergraphik 64 diese 8 Farben bereits den 16 Farben des Commodore 64 sinnvoll zu:

```
CF |00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15
---+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--
DF |00|07|02|04|06|05|04|01|03|02|03|07|07|05|04|07
```

Dabei bedeuten: CF Farbe Ihres CBM 64 und
DF zugeordnete Druckerfarbe

Sie können diese Zuordnung z.B. für Falschfarbenbilder je nach Belieben ändern mit dem Befehl:

```
- HCOPYD cf,df
```

Der Befehl zur Erstellung einer Hardcopy lautet:

```
HCOPYC#n,d
```

Wie Sie sehen, können Sie bei diesem Hardcopy-befehl den weiteren Parameter d angeben. d ist entweder 1 oder 2 und gibt die Punktdicke des Ausdrucks an. So wird bei d=1 jeder Punkt des Bildschirms als 1 Punkt auf dem Ausdruck dargestellt: bei d=2 besteht ein Bildschirmpunkt aus 2x2 Druckpunkten - entsprechend größer, aber auch gröber ist der Ausdruck. Um die Proportionen zu wahren, wird im Multicolormodus für einen doppelt breiten Punkt des Bildschirms ebenfalls ein doppelt breiter Punkt ausgedruckt.

Ein Programm zum Ausdrucken eines HGR-Bildes könnte z.B. so lauten:

```
10 GMODE 0,1  :REM ENTSPRECHENDEN GRAPHIKMODUS EIN!
20 OPEN 1,4,1 :REM SECUNDAERADRESSE 1 FUER INTERFACE
30 HCOPYD 3,2 :REM COMPUTERFARBE 3 DER DRUCKERFARBE 2 ZUORDNEN
40 HCOPYC#1,2 :REM DOPPELT GROSSE HARDCOPY
50 CLOSE 1    :REM KANAL SCHLIESSEN
```

Wichtig! Sollen MC-Graphiken ausgedruckt werden, so ist auch ein entsprechender MC-Graphikmodus einzuschalten (gleiches gilt für HGR). Sie müssen also stets in dem Modus ausdrucken, in dem das Bild gezeichnet ist. Beachten Sie auch, daß in MC die Farbe 3 im Farb-RAM durch schreiben im Text-Modus zerstört werden kann (s. GLOAD/GSAVE). Laden Sie also ein auszudruckendes Bild am besten per Programm in den Speicher und drucken dann aus!

Haben Sie beim Ausdruck etwas Geduld! Vergessen Sie nicht, daß pro Bild 256.000 Punkte, also 250 K-Byte übertragen werden müssen (bei doppelter Punktbreite).

2.14.4 Große Posterhardcopy - Die Hardcopy für jeden Drucker

Manch einem werden die Hardcopies, die durch obigen Befehl möglich sind, viel zu klein sein. Aus diesem Grunde geben wir hier ein BASIC-Programm an, das Ihnen eine große Posterhardcopy auf Papier bringt.

Das Programm erzeugt eine Hardcopy, die eine Breite von etwa 3 DIN A4-Blätter-Breiten und eine Länge von etwa 3 DIN A4-Blätter-Längen besitzt. Sie benötigt also insgesamt 9 Blätter, die nachher zusammengeklebt werden müssen (je nach Drucker und Belieben können es auch mehr oder weniger werden). Sie können damit Plakate für Ihre nächste Demo vorbereiten, Ihren Namen dick und fett über das Bett hängen (oder über den Badezimmerspiegel, falls Sie morgens Identitätsprobleme haben) oder einen 9 Seiten langen Brief an Ihre(n) Freund(in) schreiben, ohne sich viel aus den Fingern saugen zu müssen. Ich werde mir allerdings keine weiteren Gedanken darüber machen, was Sie mit dem folgenden Programm anfangen werden.

```
100 REM **********************
110 REM **                  **
120 REM **  POSTERHARDCOPY  **
130 REM **                  **
140 REM **********************
150 REM
160 PRINT CHR$(147) :REM BILDSCHIRM LOESCHEN
170 POS= 10,10
180 PRINT "BILD EINLADEN (J/N)? ";
190 GET A$ : IF A$="" THEN 190
200 PRINT A$
210 IF A$<>"J" THEN 370
220 REM
230 REM BILD EINLADEN:
240 REM **************
250 REM
260 POS= 10,12
270 INPUT "BILDNAME";N$ :REM NAMEN ERFRAGEN
280 GMODE 0,1 : SCOL= 0,0 : SCOL=1,5
290 GLOAD G,1,N$,8      :REM BILD EINLADEN
300 GMODE ,0
310 IF ST<>64 THEN:POS= 10,14 : PRINT "FEHLER!" : END
320 REM
330 REM
340 REM DRUCK INITIALISIEREN:
350 REM *********************
360 REM
370 LE$=" "     :REM STRING FUER KEIN PUNKT
```

```
380 REM STRING FUER PUNKT GESETZT:
390 PU$=CHR$(18)+" "+CHR$(146)    :REM RVS ON/RVS OFF
395 ZE$=CHR$(8)+CHR$(13)+CHR$(15):ZEILENENDE
400 XA= 107       :REM ANZAHL ZEILEN PRO SEITE
410 YA=  70       :REM ANZAHL SPALTEN PRO SEITE
420 REM
430 POS= 0,15
440 PRINT "DRUCKER EINSCHALTEN UND TASTE BETAETIGEN"
450 GET A$ : IF A$="" THEN 450
460 GMODE 0,5     :REM HGR BEFEHLIGEN
470 REM
480 REM
490 OPEN 1,4      :REM DRUCKDATEI OEFFNEN
500 REM
510 REM
520 REM DRUCK BEGINNEN:
530 REM ****************
540 REM
550 Y1=-1+YA-1
560 ZE=INT (200/YA)    :REM ANZAHL BLAETTER (BREITE) -1
570 FOR Z= 1 TO ZE     :REM BLATTSCHLEIFE
580   Y0=Y1+1          :REM STARTWERT FUER BLATT
590   Y1=Y1+YA-1       :REM ENDWERT FUER BLATT
600   IF Z=ZE THEN Y1=199
610   REM
620   FOR X=319 TO 0 STEP -1 :REM X-SCHLEIFE
630     FOR Y= Y0 TO Y1      :REM Y-SCHLEIFE
640       PLOT T TEST,4,X,Y  :REM PUNKT ABTASTEN
650       IF TEST=0 THEN PRINT#1,LE$; : GOTO 670 :REM LEERSTELLE
660       PRINT#1,PU$;                           :REM PUNKT
670     NEXT Y
680     REM
690     PRINT#1,ZE$          :REM NAECHSTE ZEILE
700     REM AUF BLATTENDE TESTEN:
710     QU=(320-X)/XA
720     IF INT(QU)=QU THEN GOSUB 850
730   NEXT X
740   REM
750   GOSUB 850              :REM NEUES BLATT
760 NEXT Z
770 REM
780 REM
790 CLOSE1 : END
800 REM
810 REM
820 REM NAECHSTES BLATT:
830 REM *****************
840 REM
850 PRINT
860 PRINT "BITTE NEUES BLATT EINLEGEN"
870 PRINT "UND TASTE BETAETIGEN!"
```

```
880 GET A$ : IF A$="" THEN 880
890 RETURN
```

Das Programm scheint Ihnen für eine Hardcopy etwas kurz? Nun ja, es funktioniert zumindest. Das Ganze resultiert aus einem einfachen Trick: Wir machen zwar eine Hardcopy von unserem Graphikbildschirm, die einzelnen Punkte werden aber nicht im Graphikmodus, sondern als ganz normale Zeichen (hier Leerzeichen und invertiertes Leerzeichen) auf dem Drucker ausgegeben. Jeder Punkt besitzt also die Größe eines Druckerzeichens. Das Prinzip ist also ganz einfach: Wir fragen nacheinander alle Graphikpunkte ab und geben für jeden Punkt ein Zeichen auf dem Drucker aus. Damit funktioniert diese Hardcopy für jeden Drucker und sogar für Typenrad- oder gar nicht graphikfähige Drucker!

Doch gehen wir das Programm doch einmal detailierter durch:

Zunächst einmal wird der normale Text-Bildschirm in Zeile 160 gelöscht, um freie Bahn zu haben. Nun haben Sie zwei Möglichkeiten: Entweder das auszudruckende Bild steht bereits in Seite 1 des Graphikbildschirms, dann geben Sie auf die Frage "Bild einladen (j/n)?" irgendeine Taste ein, nur nicht "J". Wollen Sie das Graphikbild jedoch zunächst von Diskette laden, dann drücken Sie eben dieses "J" und Sie werden nach dem Dateinamen gefragt (Zeile 270). Während das Bild nun eingeladen wird (Zeile 290) können Sie den Vorgang direkt am Bildschirm miterleben, da vorher in Graphik umgeschaltet wurde (Zeile 280). Das Bild sollte mit dem einfachen G-Sekundärbefehl auf Diskette abgespeichert vorliegen (32 Blöcke), ansonsten ändern Sie den Ladebefehl entsprechend.

Wurde nun fehlerfrei geladen, ist alles in Ordnung und es kann gedruckt werden. Ist beim Laden jedoch ein Fehler aufgetreten (z.B. File not found), so wird dies durch das Abfragen der Status-Variable ST in Zeile 310 festgestellt und angezeigt.

Nach diesem Vorspann kann es endlich zum Kern der Sache gehen. Wir beginnen mit der Druckvorbereitung: Zunächst werden die Strings festgelegt, die dem Drucker für einen nicht gesetzten Punkt LE$ (hier ein Leerzeichen; Zeile 370), für einen gesetzten Punkt PU$ (hier ein invertiertes Leerzeichen; Zeile 390) und bei einem Zeilenende übermittelt werden. Hier können Sie natürlich Ihre Phantasie spielen lassen, indem Sie z.B. alle gesetzten Punkte als Sternchen o.ä. ausgeben usw. Sie könnten z.B. auch auf den Gedanken kommen, diese Hardcopy für Multicolor-Bilder umzuschreiben und verschiedene

Grauwerte realisieren etc. Die Sequenz für ein Zeilenende schaltet den Drucker mit CHR$(8) zunächst in den Graphikmodus, was die Folge hat, daß der Zeilenabstand so gering wird, daß die einzelnen Zeilen aneinanderstoßen. Dann wird ein "carriage return" ausgegeben, um an den nächsten Zeilenanfang zu kommen. Da für uns der Graphikmodus aber völlig uninteressant ist, schalten wir ihn mit einem CHR$(15) sogleich wieder aus. Die Sequenz kann natürlich von Drucker zu Drucker verschieden sein, da ja lediglich der Zeilenabstand verringert werden soll. Bei vielen (z.B. Epson-kompatiblen) Druckern braucht dafür nur einmal am Programmanfang eine einzige Sequenz ausgegeben werden. Es sieht aber z.B. auch ganz nett aus (und vergrößert die Hardcopy), wenn Sie zwischen jeder Zeile einen weißen Streifen lassen. Vielleicht wird dann ein Punkt auch gleich zwei Zeichen breit (z.B. ein inverses und ein normales Leerzeichen). Probieren Sie den Effekt doch einmal aus, vergessen Sie aber nicht jeweils die nächsten zwei Parameter zu ändern:

In den nächsten beiden Zeilen (400/410) geben Sie an, wieviele Zeilen auf dem Drucker pro Druckseite ausgegeben werden sollen (XA) bzw. wieviele Zeichen in eine Druckzeile passen sollen. Da das ganze Bild quer ausgedruckt wird, bedeutet der erste Wert gleichzeitig, wieviel Punkte in x-Richtung, der zweite Wert, wieviel Punkte in y-Richtung auf eine Druckseite passen sollen. Hier wurden Werte gewählt, die Ihnen eine genügend große Klebefalz zum nachfolgenden Zusammenkleben frei lassen.

Gleichzeitig sind diese Werte natürlich druckerabhängig. Die hier angegebenen Werte gelten für die Drucker:

- MPS 801-803
- CBM 1525/1526
- Seikosha GP 100VC
- Epsonkompatible mit DATA BECKER-Interface im Commodore-Modus

Da es Drucker gibt, die auch im Text-Modus einen engeren Zeilenabstand einstellen können (z.B. Epson), passen nach der Einstellung dieses engen Zeilenabstandes natürlich weitaus mehr Zeilen auf eine Seite. Gleichzeitig erscheint die Graphik auch nicht mehr so gestreckt wie in dieser Version.

Doch machen wir einmal weiter: Nachdem Sie der Aufforderung den Drucker einzuschalten nachgekommen sind (Zeilen 430-450), wird in Zeile 460 der Graphikmodus eingeschaltet. Dabei bleibt (Modus 5) die Text-Anzeige sichtbar. Dies ist notwendig, um den späteren PLOT-Befehl in die HGR zu leiten.

Jetzt wird die Druckdatei geöffnet (Zeile 490). Dieser Befehl kann natürlich ebenfalls von Drucker zu Drucker und von Interface zu Interface variieren. Hierzu sollten Sie einmal in Ihrem Drucker- bzw. Interface-Handbuch nachschauen.

Ab Zeile 550 wird es nun ernst, der Druck beginnt! Zunächst wird die Anzahl der Blätter in der Breite (oder Höhe, jedenfalls in y-Richtung) minus 1 berechnet (Z. 560). Diese ergibt sich aus der Anzahl der Punkte in y-Richtung (200) und der Anzahl der Punkte pro Blatt (Zeichen pro Zeile). Dieser Wert fließt in die Blattschleife ein, die bestimmt, wieviel Blätter nachher nebeneinander liegen.

Zeilen 580/590 berechnen dann die y-Start- und Endwerte, da ja immer nur in horizontalen Streifen gedruckt werden kann.

In den Zeilen 620 und 630 beginnen die beiden Hauptschleifen, die jeweils die Koordinaten des abzufragenden Punktes in x- und y-Richtung herunter- bzw. hochzählen. Der jeweilige Punkt wird dann durch den Test-PLOT-Befehl in Zeile 640 abgefragt und entsprechend dem Ergebnis der String für einen gesetzten oder einen nicht gesetzten Punkt an den Drucker übermittelt (Z. 650/660). Die y-Schleife druckt dann jeweils eine Druckerzeile. Nach jeder Zeile wird dann geprüft, ob bereits alle Zeilen einer Seite gedruckt wurden (Zeilen 710/720). Immer wenn der Quotient QU aus der Anzahl aller bereits gedruckten Zeilen (320-X) und der Anzahl der Zeilen pro Seite eine ganze Zahl wird, wird der Integer dieses Quotienten INT(QU) gleich dem Quotienten QU selbst und damit muß eine neue Seite begonnen werden.

Zu diesem Zwecke wird in das Unterprogramm ab Zeile 850 verzweigt, das Sie zum Seitenwechsel auffordert.

Wurde ein Graphikstreifen von x=0 bis x=319 gedruckt, ist der nächste darunter liegende Streifen dran (z-Schleife). Das ganze wiederholt sich solange, bis alles fertig gezeichnet ist. Die Druckdatei wird wieder geschlossen (Zeile 790) und Sie sind wieder Herr Ihres Rechners/Druckers.

Hat Ihr Drucker nun alles in dieser Weise zu Papier gebracht, ist es Ihre Aufgabe, die einzelnen Blätter in der richtigen Weise zusammenzukleben. Haben Sie vor dem Druck das Papier im Drucker richtig justiert, brauchen Sie nichts mehr wegzuschneiden, um Absätze im Bild zu vermeiden. Sie sollten Ihren Druckkopf ganz nach links schieben (bzw. das Papier so justieren, daß der Drucker an der äußersten linken Papierkante beginnt). Gleichzeitig sollte Ihr Drucker ebenfalls an der äußersten noch machbaren oberen Papierkante beginnen.

Haben Sie die einzelnen Blätter nicht durcheinander gebracht, brauchen Sie auch nicht groß zu puzzeln, da doch in einer gewissen Ordnung gedruckt wird. Ich traue Ihnen da doch schon einiges zu. Aus diesem Grunde unterlasse ich alle weiteren Ausführungen.

Na, hervorragend, jetzt haben wir alle Graphikbefehle besprochen. Es fehlen nur noch einige nützliche Befehle der Supergraphik, die Ihnen das Leben erleichtern werden.

2.15 Noch einige nützliche Befehle der Supergraphik

Neben den Graphikbefehlen, die wir inzwischen kennengelernt haben, besitzt die Supergraphik noch eine Reihe anderer Kommandos, die die verschiedensten Aufgaben meistern. Diese Befehle sind in zwei große Gruppen zu dividieren:

- Soundbefehle
- Utilities

Befassen wir uns nun zuerst einmal mit der ersten Gruppe:

2.15.1 Soundbefehle

Unser Computer hat nicht nur phantastische Graphikeigenschaften, wir können ihm sogar recht ordentliche Klänge entlocken. Da dürfen in der Supergraphik natürlich die Befehle zur Klangsteuerung nicht fehlen. Zu diesem Zweck gibt es insgesamt 4 Kommandos, die das normale Commodore-BASIC nicht besitzt:

Dann also zu den Befehlen, die nicht nur das Herz, nein auch die Ohren erfreuen. Gemeint sind, wie soll es anders sein, die Tonbefehle.

Töne werden in der Supergraphik ebenfalls über IRQ gesteuert. Sie sind also in der Lage, richtige polyphone (3 unabhängige(!) Stimmen) Musikstücke, Soundetüden oder auch Klangeffekte zu erzeugen. Dabei ist jedes Register des SID (Sound Interface Device), also des Synthesizers in Ihrem Rechner ansprechbar. Über die Funktionen dieser Register informiert Sie Ihr Handbuch.

Falls nichts weiteres angegeben, gilt für alle Parameter der Bereich 0-15.

VOLUME=

Befehl:	VOLUME= v	
Beispiel:	VOLUME= 15	
Parameter:	v:	Lautstärke
Funktion:	Einstellen der Lautstärke	

Mit diesem Befehl stellen Sie die Lautstärke Ihres Tones ein. Er gilt für alle Stimmen, sollte am Anfang definiert werden und kann natürlich zwischendurch geändert werden.

SOUND

Befehl:	SOUND st,w,a,d,s,r (,f (,p))	
Beispiel:	SOUND 1,3,12,11,5,5	
oder	SOUND 1,3,12,11,5,5,1	
oder	SOUND 1,3,12,11,5,5,1,100	
Parameter:	st:	Stimme (1-3)
	w:	Wellenform (0-8)
	a:	Attack (Anschwellzeit)
	d:	Decay (Abschwellzeit)
	s:	Sustain (Haltelautstärke)
	r:	Release (Ausklingzeit)
	f:	Filter an/aus (0-1)
	p:	Pulsrate (Tastverhältnis)(0-4095)

Funktion: Einstellen des Klangbildes

Mit diesem einen Befehl stellen Sie das Klangbild eines Tones einer Stimme (st) ein. ür w sind die Werte 0-8 zulässig. Sie bestimmen folgende Funktionen (werden mehrere Wellenformen gleichzeitig

angegeben, so ist jeweils die logische UND-Verknüpfung dieser beiden Wellenformen gemeint):

w=0: Stimme ist stumm
w=1: Dreiecksschwingung
w=2: Sägezahn
w=3: Dreieck und Sägezahn
w=4: Rechteck
w=5: Dreieck und Rechteck
w=6: Sägezahn und Rechteck
w=7: Dreieck, Sägezahn und Rechteck
w=8: Rauschen

Bei Bedarf können Sie mit f die Filter für diese Stimme ein (f=1) oder aus (f=0) schalten (dabei kann ein hardwarebedingtes Knacken erfolgen).

Weiterhin können Sie noch die Pulsrate der Rechteckschwingung mit p festlegen (dies hat natürlich nur Effekt, wenn die Rechteckschwingung ausgewählt ist: w=4; w=6; w=7), wobei für p Werte von 0 bis 4095 zulässig sind.

FILTER

Befehl:	FILTER fa (,ff,rf)	
Beispiel:	FILTER 4	
oder	FILTER 4,1000,14	
Parameter:	fa:	Filterart (0-7)
	ff:	Filtergrenzfrequenz (0-2047)
	rf:	Filterresonanzfrequenz (0-15)
Funktion:	Einstellen der Filter	

Mit FILTER bestimmen Sie den Filterzustand des SID, der ja bekanntlich für alle Stimmen identisch ist.
fa ist die Filterart, wobei gilt:

fa=0: kein Filter an
fa=1: Tiefpass
fa=2: Bandpass
fa=3: Tief- und Bandpass
fa=4: Hochpass
fa=5: Hoch- und Tiefpass (Bandsperre)
fa=6: Hoch- und Bandpass
fa=7: Hoch-, Band- und Tiefpass (Bandsperre)

Optional können Sie weiterhin die Filtergrenzfrequenz (ff) (für den Fall des Bandpassfilters) und die Filterresonanzfrequenz (rf) setzen.

TUNE

Befehl:	TUNE st,l,n,o (,v)	
Beispiel:	TUNE 2,30,3,4	
oder	TUNE 2,30,3,4,14	
Parameter:	st:	Stimme (1-3)
	l:	Länge des Tones (0-65535)
	n:	Note (0-12)
	o:	Oktave (0-7)
	v:	Verstimmung (0-255)
Funktion:	Spielen eines Tones	

Jetzt haben Sie alles vorbereitet; dann kann es ja los gehen:
Mit TUNE spielen Sie einen Ton mit den eben definierten Eigenschaften in der Stimme st.

l gibt die Länge des Tones in 1/60 Sekunden wieder, d.h. Ihr Ton wird (l+1)/60 Sekunden lang gehalten.

Mit n bestimmen Sie die Note, die gespielt werden soll und gibt die Position des Tones in der Tonleiter an (Ton c=1):

```
c ,cis, d ,dis, e , f ,fis, g ,gis, a ,ais, h
1   2   3   4   5   6   7   8   9  10  11  12
```

(ist n=0, so wird das h der darunter liegenden Oktave gespielt).

o schließlich definiert die Oktave. Zur Orientierung: der Kammerton a (440 Herz) besitzt die Werte: n=10;o=4.

Wenn Sie wünschen, können Sie Ihre Töne mit v etwas verstimmen, sodaß z.B bei einem Dreiklang ein vollerer Klang entsteht. v kann dabei Werte von 0-255 annehmen:

v= 1-127: Verstimmung um v nach oben
v= 255-128: Verstimmung um 256-v nach unten

Hier ein kleines Beispiel einfachster Klangprogrammierung, das sich als Schema für die Liedererzeugung verwenden läßt:

```
10 VOLUME=15
20 SOUND 1,.... : SOUND 2,..... : SOUND 3,.....
30 FILTER ..... : READ Z : REM ZAHL DER TOENE
40 FOR X=1 TO Z : READ S,L,T,O : TUNE S,L,T,O : NEXT X
90 DATA Z,.......Lieddefinition........
```

In DATA wird zu Anfang die Zahl der Töne angegeben, die gespielt werden sollen (Z). Dann folgen Angaben zu den zu spielenden Tönen (in der Reihenfolge des Tonablaufes) mit folgendem Format:

Stimme, Länge, Ton, Oktave

So schnell lernen Sie, ein Instrument zu spielen!

2.15.2 Utilities für alle Zwecke

Supergraphik 64 besitzt nicht nur Graphik- und Tonbefehle, sondern unterstützt gleichfalls mit einigen wesentlichen Kommandos Ihre Programmierarbeit. Sie werden sehen, daß hier eine Auswahl getroffen wurde, die die aus eigener Erfahrung notwendigsten Funktionen vereint. Selbstverständlich gibt es noch eine Reihe weiterer Programmierhilfen, doch sind die oft relativ unbrauchbar und platzverschwendend.

DIRECTORY

Befehl:	DIRECTORY
Beispiel:	DIRECTORY
Parameter:	---
Funktion:	Anzeigen des Disketteninhaltsverzeichnisses ohne Programmverlust

Der größte Mangel des originalen BASIC offenbart sich wohl mit dem Fehlen eines DIRECTORY-Befehls. Diese Lücke soll hier nun geschlossen werden. Der Befehl ist recht einfach zu bedienen: Eingeben, (return) drücken, fertig. Ein im Speicher befindliches Programm wird dadurch nicht gelöscht!

MERGE

Befehl:	MERGE "filename",ga
Beispiel:	MERGE "HUSTENBONBON",8
Parameter:	ga: Geräteadresse
Funktion:	Anhängen eines BASIC-Programmes von Diskette an ein im Speicher befindliches

Mit diesem Befehl ist es Ihnen möglich, an ein BASIC-Programm im Computer ein weiteres anzuhängen, das Sie durch MERGE einladen. Damit können Sie sich eine kleine Routinensammlung auf einer Diskette anlegen, die sie bei Bedarf zusammenladen. Oder Sie holen die DATA-Zeilen für verschiedene Sprites in Ihr Programm.

Die Zeilennummern sollten sich jedoch, falls in dem eingeladenen Programm irgendwelche Zeilenansteuerungen (GOTO ...) vorhanden sind, nicht mit den Nummern des Vorprogrammes überschneiden, da es ansonsten (auch nach RENUMBER) Konfusionen gibt.

Vorsicht bei sehr langen Programmteilen! Reicht der Speicher nicht aus, um beide Programme zu fassen, so werden Teile der Supergraphik überschrieben, was unmittelbar zu Absturz führt! Speichern Sie also bei langen Programmen vorher grundsätzlich das im Speicher befindliche Programm auf Diskette (Kassette) ab!

RENUM

Befehl:	RENUM sz,a
Beispiel:	RENUM 10,50
Parameter:	sz: Startzeile der neuen Version
	a: Abstand der BASIC-Zeilen (1-255)
Funktion:	Umnumerierung eines BASIC-Programmes

Ein weiteres Juwel unter den Programmierhilfen ist der bekannte RENUMBER-Befehl. Mit ihm ist es möglich, die Zeilennummern eines ganzen BASIC-Programmes umzubenennen. Dabei werden selbstverständlich alle GOTOs, GOSUBs, ON...GOTOs, ON...GOSUBs, THEN...s, RUNs und LISTs mit verändert, sofern eine Zeilennummer folgt (auch z.B. LIST 40-70 etc.). Dadurch schaffen Sie problemlos Platz zwischen zwei Zeilen, wenn Sie etwas einfügen wollen. Das funktioniert natürlich auch bei Programmen, die nicht mit der Supergraphik geschrieben wurden.

Tauchen innerhalb des Programmes Zeilen in GOTOs etc. auf, die es eigentlich gar nicht gibt, so wird dort die Zahl 65535 eingesetzt.

Der Befehl DTASET wird nicht umnumeriert, da dessen Zeilennummer auch berechnet werden kann (indirekt). Hier müssen also als einziges von Hand Änderungen vorgenommen werden.

Bei Programmen, bei denen der gewählte Zeilenabstand und die Startzeile dazu führen würden, daß Zeilennummern größer als 63999 auftauchen würden, wird vor der Umnumerierung die Meldung "ILLEGAL QUANTITY ERROR" ausgegeben, um Ihr Programm vor Zerstörung zu sichern. Ein BASIC-Programm, das mit RENUM behandelt wird, kann in seiner Länge zu oder abnehmen, da die Zeilennummern hinter GOTOs etc. länger oder kürzer werden können! Programme, die mit ihrer Länge rechnen, sollten dies beherzigen.

KEY

Befehl:	KEY ft,"definition"
Beispiel:	KEY 2,"LIST"+CHR$(13)
Parameter:	ft: angewählte Funktionstaste (1-8)
	"definition": 16 Zeichen-Wort
Funktion:	Umprogrammieren der Funktionstasten

Einfach wird das Programmieren gleichfalls durch die Belegung der 8 Funktionstasten Ihres Rechners. Damit genügt ein Tastendruck, um ganze Befehle in Aktion zu bringen. Die in "definition" stehenden Buchstaben werden von nun an der mit ft angegebenen Funktionstaste zugeordnet. Bei dem Druck auf diese Taste wird dann automatisch diese Zeichenfolge (maximal 16 Buchstaben) auf den Bildschirm geschrieben und evt. ausgeführt.

Programmieren Sie eine Taste mit einem Befehl (z.B. LIST) und wollen Sie, daß dieser Befehl direkt ausgeführt wird, dann müssen Sie noch zusätzlich den ASCII-Wert 13 (für <return>) anhängen:

```
KEY 1,"LIST"+CHR$(13)
```

Wichtig ist, daß der Druck auf eine Funktionstaste nur direkt am Zeilenanfang, direkt nach der Eingabe von <return> eine Wirkung besitzt. Sollte also die Betätigung einer Funktionstaste keinen Erfolg haben, so drücken Sie einmal kurz auf <return> und anschließend wieder auf die Funktionstaste. Jetzt sollten keine Probleme mehr auftauchen.

Nach dem Starten der Supergraphik sind die 8 Tasten sinnvoll bereits auf die folgende Art und Weise vorbelegt, um Ihnen Arbeit zu ersparen (die Belegung kann natürlich jederzeit geändert werden):

```
f1: "LIST"+CHR$(13)          || f2: CHR$(146)+"LIST"+CHR$(13)
f3: "DIRECTORY"+CHR$(13)     || f4: "GMODE,7,161,250"+CHR$(13)
f5: "GMODE,0"+CHR$(13)       || f6: "GMODE,8,161,250"+CHR$(13)
f7: "GMODE,1"+CHR$(13)       || f8: "RENUM10,10"
```

Wenn Sie also nach dem Laden z.B. auf die Taste f3 drücken, erscheint das Inhaltsverzeichnis Ihrer Diskette auf dem Bildschirm etc.

Am besten, Sie schreiben sich Ihre eigene Belegung auf, um stets die Übersicht zu behalten. Wollen Sie ständig mit Ihrer eigenen Funktionstastenbelegung arbeiten, so erstellen Sie vielleicht ein kleines Programm mit den 8 Definitionen und speichern es auf Ihre Arbeitsdiskette ab. So können Sie es jederzeit abrufen und durch ein simples RUN sämtliche Tasten programmieren!

DTASET

Befehl:	DTASET zn
Beispiel:	DTASET 100
Parameter:	zn: Zeilennummer
Funktion:	Setzen des DATA-Zeigers auf eine bestimmte Zeile

Dieser Befehl erleichtert Ihre Arbeit mit den vielen verschiedenen Spritedefinitionen stark, die bekanntlich zweckmäßigerweise in DATA-Zeilen abgelegt werden. Mit DTASET wird nämlich der Befehl RESTORE erweitert. RESTORE setzt den DATA-Zeiger wieder auf den Programmanfang. Wollen Sie also ein Element mitten aus dem Programm entnehmen, so mußten Sie früher vorher alle vorgelagerten Daten auslesen. Mit DTASET dagegen haben Sie also nun freien Zugriff (random access), indem es die Zeile bestimmt, ab der das nächste DATA-Element bei einem READ oder SREAD gelesen werden soll.

PADDLE

Befehl:	PADDLE var,pn
Beispiel:	PADDLE A,2
Parameter:	var: Zielvariable
	pn: Paddlenummer (1-4)
Funktion:	Auslesen der A/D-Wandler (Paddle-Werte)

Unterschätzen Sie diesen Befehl nicht! Ihr Rechner besitzt 4 eingebaute sogenannte Analog/Digital - Wandler. Diese elektronischen Details ermöglichen uns, Widerstände von 200 Ohm bis 200 Kiloohm zu messen. Die verschiedenen Widerstandswerte werden in Werte von 0 bis 255 übersetzt, die dann durch Software ausgelesen werden können. Anzugeben ist lediglich die Paddlenummer (1-4) - die Analog-Eingänge sind an den Eingangsports verfügbar (s. Handbuch -

Anhang) - und der Speicher (var) in den der ausgelesene Wert abgelegt werden soll.

Zum einen können durch diesen Befehl handelsübliche Paddles (Drehknöpfe) ausgelesen werden, zum anderen sind aber auch verschiedene elektronische Sensoren wie Photowiderstände, Heißleiter, Kaltleiter usw. anschließbar.

Eine besonders interessante Anwendung stellt der Gebrauch des Koala-Pad, also des Zeichentabletts dar, das eigentlich für das bekannte Zeichenprogramm konzipiert wurde. Doch durch den Befehl PADDLE ist es Ihnen gleichfalls durch ein kleines Programm möglich, das Graphiktablett zu bedienen und mit der Supergraphik zu zeichnen! Wenn Sie ein Graphiktablett besitzen, dann probieren Sie es doch einmal:

```
10 GMODE 0,1 : GCLEAR
20 PADDLE X,1 : PADDLE Y,2
30 PLOT 0,X,Y
40 GOTO 20
```

Das genügt allein schon, um einfache Zeichnungen via Graphik-Tablett anzufertigen! - übrigens auch mit normalen Paddles. (natürlich müssen Sie das Graphiktablett (Paddles) in den richtigen Port gesteckt haben und mit dem Zeichengriffel auf der Fläche malen!)

3. Anwendungen

Wir haben nun das große Kapitel der Befehlsvorstellungen der Supergraphik hinter uns gebracht, in dem uns die verschiedenen Möglichkeiten des Zeichnens von Punkten, Linien und Kreisen, der Sprite- oder sonstigen Darstellung, sowie die diversen Ausgabebedienungen und vieles mehr dargelegt wurden.

Doch was fangen wir mit diesem Wissen an? Wie verwendet man etwa Linien und Kreise, um unterschiedliche Sachverhalte darzustellen? Was bringen uns die Sprites? Und so weiter und so fort.

In diesem Kapitel sollen Ihnen einige Beispiele nahegebracht werden, die Ihnen die Arbeit mit diesen Themen schmackhaft machen sollen. Viele Tricks und Tips können Sie den einzelnen Paragraphen entnehmen, die Sie in Ihren Programmen gut verwenden können.

Drei große Abschnitte behandeln die verschiedenen Möglichkeiten der hochauflösenden Graphik oder einer schönen Anwendung der Sprites in Laufschriften, die besonders im kommerziellen Gebrauch Verwendung finden. Stellen Sie sich doch einmal die Repräsentation eines Produktes oder Sonderangebotes mit Hilfe der Sprites und den enormen Graphik- und Soundfähigkeiten des Commodore 64 vor. Sind das nicht verlockende Aussichten?

Als letzten Abschnitt zeigen wir Ihnen die Möglichkeiten, die sich z.B. bei Spielen ergeben, einige nützliche Routinen werden auch hier vorgestellt.

3.1 Graphikanwendungen

Lassen wir keine Müdigkeit aufkommen, gehen wir gleich zur Sache. Alle im folgenden angeführten Beispiele verwenden den Befehlssatz der Supergraphik oder im Falle des Kuchendiagrammes die Befehle, die Ihnen durch das kleine Graphik - Paket in Kapitel 5 zur Verfügung gestellt werden.

Vor allem die hochauflösende Graphik läßt sich gut für kommerzielle Zwecke, aber auch für hübsche Privatanwendungen (Schule, Beruf, Freizeit) verwenden. Arbeiten Sie einmal die folgenden Abschnitte durch.

3.1.1 Funktionendarstellung

Eine beliebte Art, die hochauflösende Graphik zu nutzen, ist das Zeichnen der Graphen verschiedener Funktionen. Dabei werden oft in Beispielprogrammen Sinus- oder Cosinuskurven verwendet (s. Beispielprogramm in Paragraph 5.2.2.1). Dies hat meist zwei Gründe: erstens sieht das Ganze recht eindrucksvoll aus, und zweitens - was wohl das Wichtigere ist - man hat nicht so große Probleme mit eventuellen Bereichsüberschreitungen. Letzteres ist die größte Schwierigkeit bei dem Zeichnen solcher Bilder. Um diesem Problem auf die Spur zu kommen, müssen wir uns wieder ein wenig mit der Mathematik beschäftigen.

Eine Funktion ist eine (eindeutige) Abbildung einer Menge auf eine andere, d.h. jedem Element der einen Menge ist genau ein Element der anderen zugeordnet (aber nicht unbedingt umgekehrt). Die Elemente der ersten Menge nennt man in algebraischen Funktionen auch x, die der zweiten Menge y. x und y sind in diesem Fall durch irgendeine mathematische Formel bzw. Gleichung verbunden (in dem Fall der Linien aus Abschnitt 5.2.2.2 beispielsweise durch die Geradengleichung):

$y = f(x)$ sprich: y gleich f von x

Will man z.B. ein Zuordnungspaar feststellen, so setzt man für x einen beliebigen Wert ein und kann so y errechnen. Die Beziehung zwischen x und y kann man gleichfalls graphisch in einem Koordinatensystem darstellen. Dabei stellt die horizontale Achse des Systems alle möglichen x- (Abszisse), die vertikale dagegen alle möglichen y-Werte (Ordinate) der Funktion dar. Kennen wir ein x/y-Paar, so fällen wir ein Lot auf den betreffenden x-Wert der x- und entsprechend auf den y-Wert der y-Achse. Am Schnittpunkt dieser beiden Senkrechten wird nun ein Punkt eingetragen. Um ein möglichst genaues Bild des entstehenden Graphen zu erhalten, müssen wir viele solcher Koordinatenpaare berechnen und in unser System eintragen. In der Computerpraxis sieht das dann so aus, daß schrittweise x-Koordinaten, die stetig von einem bestimmten Wert bis zu einem zweiten Wert ansteigen (durch eine FOR...NEXT-Schleife realisierbar), in die jeweilige Formel eingesetzt werden, und dadurch die fehlende y-Koordinate errechnet wird. Das Koordinatenpaar dient nun zum Zeichnen eines Punktes auf dem Bildschirm. Beim Zeichnen einer Linie oder eines Kreises im 5. Kapitel wird übrigens das gleiche Prinzip angewendet:

```
100 REM ************************
110 REM **                    **
120 REM **  QUADRATFUNKTION   **
130 REM **                    **
140 REM ************************
150 REM
160 GMODE 0,1 : GCLEAR
170 SCOL= 0,6 : SCOL= 1,1
180 FOR X=0 TO 319
190 Y = X^2 : REM FUNKTIONSWERT ERRECHNEN
200 IF Y>199 THEN WAIT 198,255 : GMODE ,0 : END : REM BEREICHSUEBERSCHRE
ITUNG
210 PLOT ,X,Y : REM PUNKT SETZEN
220 NEXT X
```

Wie Sie sehen wird hier in Zeile 200 geprüft, ob der y-Wert noch im Bereich des Bildschirms liegt, um ein ILLEGAL QUANTITY ERROR zu verhindern. Unter anderem darüber soll hier diskutiert werden.

Die in diesem Programm gewählte Funktion bereitet uns ein wenig Kopfzerbrechen, denn zum ersten kommt nur einer der erwarteten zwei Parabeläste auf das Bild und der auch noch verkehrt herum, dann ist die Funktion durchaus nicht bildfüllend, und drittens wurden viel zu wenig Punkte berechnet, um einen schönen Kurvenverlauf zu erreichen. Dies hat die folgenden Gründe:

Auf unserem Computer besitzt unser Koordinatensystem eine festgelegte Ausdehnung. Wir können für x und y beispielsweise keine negativen oder gebrochenen Zahlen oder Werte einsetzen, die größer sind als 319 (für x) bzw. 199 (für y). Trotzdem gibt es Funktionen, die gerade in diesen Bereichen die interessanten Teile besitzen. So liefern einfache Winkelfunktionen wie:

$y = \sin(x)$ oder $y = \cos(x)$

nur Werte zwischen -1 und 1 für y und besitzen eine Schwingungsperiode von 2*pi, also ca. 6, was direkt für unsere Anwendung nicht zu gebrauchen ist.

Es existieren nun Methoden, die diesem Dilemma Abhilfe verschaffen. Dazu gehören:

- Scalierung
- Verschiebung
- Verzerrung

a) Scalierung:

Unter Scalierung versteht man das Vergrößern oder Verkleinern der einzelnen Koordinatenwerte und damit der gesamten Kurve. Dies wird durch einfaches Multiplizieren der x- und(!) y-Werte mit einem bestimmten Faktor erreicht. Dies kann auf zwei Arten geschehen:

1.) Es werden zunächst die normalen Werte für x zur Berechnung der unveränderten Funktion verwendet und nachher x- und y-Wert mit dem besagten Faktor multipliziert. Dabei hängt es vom Faktor (hier f genannt) ab, ob Sie damit den Graphen verkleinern oder vergrößern:

Faktor	Auswirkung
$0<f<1$	Verkleinerung
$f>1$	Vergrößerung

Wählen Sie zusätzlich $f<0$, so erscheint die Funktion spiegelverkehrt und kopfüber. Eine BASIC-Routine wäre hierzu etwa:

```
110 F=30 : REM VERGROESSERUNGSFAKTOR
120 FOR X=0 TO 10
130 Y = SIN(X) : REM WERT ERRECHNEN
140 Y = F*Y : X = F*X : REM SCALIEREN
150 IF Y>199 OR X>319 THEN WAIT 198,255 : SYS OF : END
160 PLOT ,X,Y : REM PUNKT ZEICHNEN
170 NEXT X
```

Dieses Programm läuft natürlich nur mit der Supergraphik. Außerdem liefert es noch negative Werte für y.

2.) Sie bauen die Scalierung direkt in die Funktion mit ein. Dazu erzeugen Sie direkt scalierte Werte für x (durch die FOR...NEXT-Schleife), müssen diese in der Formel dann aber durch Dividieren von x durch f wieder rücksetzen. Die y-Scalierung erfolgt dann direkt in der Formel. Das könnte dann etwa so aussehen:

```
110 F=30 : REM VERGROESSERUNGSFAKTOR
120 FOR X=0*F TO 10*F
130 Y = F * SIN(X/F) : REM SCAL. WERT
140 IF Y>199 THEN WAIT 198,255 : SYS OF : END
150 PLOT ,X,Y : REM PUNKT ZEICHNEN
160 NEXT X
```

Diese Form hat insbesondere drei Vorteile: Erstens ist sie sichtbar kürzer und schneller, zweitens ist damit gleich von Anfang an überprüfbar, ob größere Werte für x gewählt werden, als erlaubt, denn wir wählen den Umfang der Schleife direkt danach und drittens berechnen Sie schon hier statt der obigen 11 ganze 301 Werte, was die Genauigkeit der Kurve natürlich beträchtlich steigert. Trotzdem hat die ganze Sache noch einen Haken: y wird immer noch negativ.

b) Verschiebung:

Um diesen Punkt zu beheben, können wir den gesamten Graphen in dem Koordiantensystem in alle Richtungen verschieben. Wir können also die Sinuskurve aus dem negativen herunter in den positiven Bereich verschieben, so daß y nur noch positive Werte annimmt. Oder wir verschieben den obigen Quadratfunktionsgraphen um einen bestimmten Betrag nach rechts, wodurch sein linker Ast sichtbar wird.

Dies funktioniert, indem zu den beiden Koordinaten jeweils bestimmte Werte hinzuaddiert werden. Diese Werte müssen nicht unbedingt gleich sein, wie bei der Scalierung, um das Aussehen des Graphen nicht zu beeinflussen. Addieren wir einen Wert b zu der y-Koordinate, so verschieben wir den Graphen nach oben bzw. nach unten, wenn b negativ ist (wir führen also eigentlich eine Subtraktion durch). Geschieht dies mit der x-Koordinate (a wird addiert), so resultiert eine Rechts- (für a>0) bzw. eine Linksverschiebung (a<0). Auch hier gibt es wieder die beiden Möglichkeiten, die völlig analog zu oben funktionieren:

1.) Der Wert der beiden Koordinaten wird errechnet und dann die beiden Summanden hinzuaddiert. Die Zeilen 130/140 in dem unter a)1.) stehenden Programm währen also so zu ersetzen (ohne Scalierung):

```
130 Y = SIN(X)
140 Y = Y+B : X = X+A
```

2.) Auch hier können wir das Ganze in eine Formel packen und erhalten (Zeilen 120/130 des unter a)2.) stehenden Programms):

```
120 FOR X=1 TO 10
130 Y = SIN(X-A) + B
```

Sie sehen, A wird von X abgezogen, statt addiert.

Unser Quadratprogramm vom Anfang könnten wir dann umschreiben, indem wir die Zeile 190 ersetzen durch z.B.:

```
190 Y = (X-13)^2 + 5
```

Schon erhalten wir ein ganz anderes Ergebnis. Hier wurde a=13 und b=5 gesetzt. Wählen wir z.B. für a größere Werte, so wird nichts gezeichnet, da y dann zu groß wird, was ja von unserer Abfrage in Zeile 200 abgefangen wird. Wie dieses Problem zu lösen ist, wird später erläutert.

c) *Verzerrung:*

Die Verzerrung ist einfach nur ein allgemeinerer Fall der Scalierung. Hier nämlich brauchen x und y nicht unbedingt mit dem gleichen Faktor multipliziert werden (wir nennen die beiden Faktoren jetzt einfach f1 und f2). Dadurch werden aber die Proportionen der Kurve verändert, d.h. es kommt zu einer Stauchung (für $0<f1/2<1$) oder Streckung ($f1/2>1$) in x- (für f1) bzw. y-Richtung (für f2). Jetzt erst werden die meisten Kurven wirklich zeichenbar. Ersetzen Sie doch einmal Zeile 190 des Quadratfunktions - Programmes durch:

```
190 Y = 0.01 * ((X-139)/1)^2 + 5
```

Sie werden strahlen, das war es. Hier werden Verzerrung und Verschiebung gleichzeitig angewandt, was zu diesem hübschen Ergebnis führt. Die einzelnen Veränderungswerte lauten: f1=1; f2=0.01; a=139; b=5 (statt mit 0.01 zu multiplizieren, könnten wir der Einfachheit halber auch durch 100 dividieren).

Doch es sind noch ein paar Dinge zu klären. Unsere Parabel steht immer noch auf dem Kopf und zweitens werden bei der Veränderung der verschiedenen Parameter immer noch ILLEGAL QUANTITY ERRORs produziert. Zunächst zum Ersten:

Dieses Phänomen taucht auf, da unser Koordinatensystem auf dem Kopf steht. Unser Nullpunkt liegt nicht unten, sondern links oben in der Ecke. Die y-Werte werden nach unten hin nicht immer kleiner, sondern größer. Diese Eigenart kann durch einen einfachen Trick be-

hoben werden. Wir drehen die Kurve einfach um, indem wir das Vorzeichen aller y-Werte umkehren (s.o.). Da wir dann aber oft nur negative Werte für y bekommen, verschieben wir die Kurve gleichfalls noch um einen bestimmten Betrag nach unten (eigentlich oben), indem wir z.B. 195 addieren. Das sähe dann so aus:

```
190 Y = - 0.01 * ((X-139)/1) + 195
```

Das Ergebnis bestätigt das oben Gesagte. Doch nun zu den Fehlermeldungen, die inzwischen immer häufiger werden. Die Ursache liegt einfach in unserer unvollkommenen Bereichsüberprüfung. Zum einen testen wir gar nicht, ob y negativ wird, zum anderen beenden wir gleich unser Programm, wenn es zu einer Überschreitung gekommen ist. Wenn Sie das obige "Quadrat"-Programm ab der Zeile 180 wie folgt ändern, dann werden Sie für alle Einsetzungen von a,b,f1 und f2 und für jede Funktion zufrieden sein, sofern sie überhaupt in dem gewählten Bereich liegt:

```
180 F1 = 1 : F2 = 0.1 : A = 160 : B = 100
190 FOR X=0 TO 319
200 Y = - F2 * ((X-A)/F1)^2 + B
210 IF Y>199 OR Y<0 THEN NEXT X : GOTO 270
220 PLOT ,X,Y : REM PUNKT SETZEN
230 NEXT X
240 WAIT 198,255 : END
```

Bevor der Computer etwas zeichnet, kann er schon eine ganze Menge Werte durchlaufen haben. Warten Sie also etwas, bevor Sie das Programm abbrechen. In diesem Programm wurde der Nullpunkt unseres verschobenen Koordinatensystems nach 160,100, also in die Mitte des Bildschirms gebracht. Wir könnten dort also auch z.B. zur Veranschaulichung die Achsen einzeichnen.

Eine weitere Möglichkeit ist beispielsweise die Niederlegung der Funktion in einer sogenannten Funktionsdefinition. Dies ist eine besondere Eigenschaft des BASIC und wird durch den Befehl DEF FN realisiert. Mit diesem Befehl wird eine Funktion definiert, die irgendwo im Programm beliebig aufgerufen werden kann, ohne sie umständlich nieder zu schreiben. Sie müßten dazu lediglich folgende Änderungen vornehmen:

```
185 DEF FN F(X)= - F2 * ((X-A)/F1)^2 + B
200 Y = FN F(X)
```

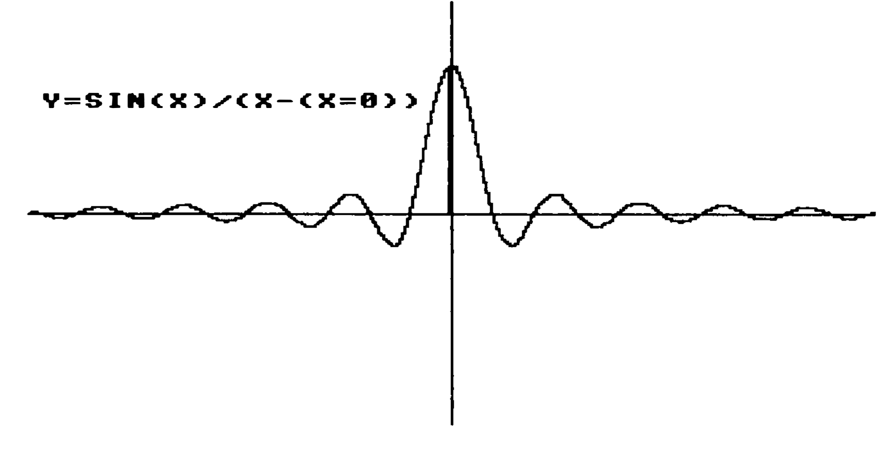
Oder wenn Sie lediglich die reine Funktion in der Definitionszeile stehen lassen wollen:

```
185 DEF FN F(X)= X^2
200 Y = - F2 * FN F((X-A)/F1) + B
```

Der jeweilige Wert für die verschiedenen Variablen wird dabei stets in die Formel eingesetzt. So können Sie irgendwo die Funktion verändern, ohne in die Schleife eingreifen oder diese suchen zu müssen.

Das folgende Programm geht noch etwas weiter. Es zeichnet für Sie eine beliebige Funktion. Sie können in eine INPUT - Abfrage Ihre Funktion eintippen und das Programm entwickelt für Sie daraus durch EinPOKEn entsprechender Bytes in einen freien BASIC-Bereich die Funktionsanweisung. Sie brauchen diese relativ komplizierte Routine nicht unbedingt zu verstehen, Sie demonstriert Ihnen aber den guten Nutzen des DEF FN - Befehls. Die Funktion wird ohne Verzerrungs- oder Verschiebungsparameter, also in "reiner" Form eingegeben. Diese werden später in der Hauptschleife in der Weise hinzugefügt, wie in den zuletzt gezeigten zwei Zeilen. Doch hier das Programm.

Wenn Sie nicht die abgespeicherten Version der Diskette benutzen, sondern eine geänderte schreiben wollen, so achten Sie bitte unbedingt darauf, in dem ersten Teil bis zum Ende der DEF FN - Zeile (also bis einschließlich Zeile 350) genau den Wortlaut mit REMs und mit genau der gleichen Leerzeichenanzahl abzuschreiben, da das Programm damit rechnet! Beachten Sie das nicht, so kommt es im Zweifelsfall zum Absturz des Programms. Vor dem ersten Start sollten Sie das Programm daher auf jeden Fall zunächst einmal abspeichern!



```
100 REM ***************************
110 REM **                       **
120 REM **  FUNKTIONENPLOTTER    **
130 REM **                       **
140 REM ***************************
150 REM
160 REM
170 REM DEF FN - ROUTINE:
180 REM ******************
190 REM
200 RESTORE:DIM A$(25):PRINT CHR$(147)
210 PRINT "GEBEN SIE EINE BELIEBIGE FUNKTION EIN :":PRINT:PRINT:PRINT
220 GOSUB 380 : REM EINGABEROUTINE
230 AD=2684-1:REM POKE-START
240 FOR X=1 TO 25:READ A$(X):NEXT X:REM DATEN EINLESEN
250 FORX=1TOLEN(A$):AD=AD+1
260 Q=X:FORY=1TO25:B$=A$(Y):FORZ=1TOLEN(B$):C$=MID$(A$,X,1)
270 IFC$=" "THENAD=AD-1:NEXTX:GOTO310
280 IFC$=MID$(B$,Z,1)THENX=X+1:NEXTZ:POKEAD,Y+169:X=X-1:NEXTX:GOTO 310
290 X=Q:NEXTY
300 POKEAD,ASC(C$):NEXTX
310 POKEAD+1,58:POKEAD+2,143
320 REM FREIHALTER:
330 DEFFNF(X)=0000000000000000000000000000000000000000000000000000000
00000000
340 REM 25 FUNKTIONEN:
350 DATA +,-,*,/,^,AND,OR,<,=,>,SGN,INT,ABS,USR,FRE,POS,SQR,RND,LO G,EXP
360 DATA COS,SIN,TAN,ATN,PEEK
370 GOTO 390: REM WEITER
380 INPUT "F(X)=";A$ : RETURN : REM EINGABEROUTINE(ZUM AUSBAU)
390 :
400 REM *** LOESCHFLAG SETZEN ***
410 :
420 POKE 53247,0
430 :
440 REM
450 REM       **************
460 REM  **** HAUPTPROGRAMM ****
470 REM       **************
480 REM
490 PRINT : PRINT "GEBEN SIE DIE FOLGENDEN PARAMETER EIN:" : PRINT
500 INPUT "F1 (X-ZERRFAKTOR)  ";F1
510 INPUT "F2 (Y-ZERRFAKTOR)  ";F2
520 INPUT "A  (X-VERSCHIEBUNG)";A
530 INPUT "B  (Y-VERSCHIEBUNG)";B
540 GMODE 0,1 : SCOL= 0,16: SCOL= 1,5:REM INITIALISIEREN
550 IF PEEK(53247)=0 THEN:GCLEAR     :REM GRAPHIK LOESCHEN
560 POKE 53247,0                     :REM LOESCHFLAG
570 REM
```

```
580 REM ACHSEN ZEICHNEN:
590 REM *****************
600 IF A>=0 AND A<320 THEN:PLOT ,A,0 TO A,199 :REM X-ACHSE
610 IF B>=0 AND B<200 THEN:PLOT ,0,B TO 319,B :REM Y-ACHSE
620 REM
630 REM ZEICHENROUTINE:
640 REM ****************
650 FL% = 1 : REM AUSSERHALB-FLAG
660 FOR X=1 TO 319
670 Y=-F2*FNF((X-A)/F1)+B
680 IF Y<0 OR Y>199 THEN FL%=1:NEXT X:GOTO 720
690 IF FL%=0 THEN:PLOT,X1,Y1 TOX,Y
700 FL%=0
710 X1=X:Y1=Y:NEXT X :REM LETZTE KOORD. MERKEN
720 POKE 198,0 : WAIT 198,255 : GET A$ : GMODE ,0 :REM GRAPHIK AUS
730 REM *****  MENUE:  *****
740 PRINT "WOLLEN SIE:" : PRINT : PRINT
750 PRINT "(1) ANDERE PARAMETER"
760 PRINT "(2) ANDERE FUNKTION"
770 PRINT "(3) GRAPHIK NICHT LOESCHEN"
780 PRINT "(4) GRAPHIK SPEICHERN"
790 PRINT "(5) GRAPHIK LADEN"
800 PRINT "(6) HARDCOPY"
810 PRINT "(7) BEENDEN"
820 WAIT 198,1 : GET A$
830 ON VAL(A$) GOTO 480,850,860,880,970,1030,870
840 GOTO 820            :REM FEHLEINGABE
850 RUN                 :REM ANDERE FUNKTION
860 POKE 2,1 : PRINT "OK" : PRINT : GOTO 820 :REM FLAG SETZEN
870 END                 :REM BEENDEN
880 REM
890 REM GRAPHIK SPEICHERN:
900 REM
910 INPUT "FILENAME,GA";FI$,GA
920 GSAVE GC,1,FI$,GA :REM SPEICHERN
930 PRINT "OK" : GOTO 820
940 REM
950 REM GRAPHIK LADEN:
960 REM
970 INPUT "FILENAME,GA";FI$,GA
980 GLOAD GC,1,FI$,GA :REM LADEN
990 GMODE 0,1 : GOTO 720
1000 REM
1010 REM HARDCOPY:
1020 REM
1030 PRINT "DRUCKER FERTIG MACHEN UND TASTE DRUECKEN"
1040 OPEN 1,4 : HCOPY#1 : CLOSE 1 :REM HARDCOPY
1050 PRINT "OK" : GOTO 820
```

Beachten Sie bitte, daß auch dieses Programm für die Supergraphik geschrieben wurde und nur nach dem Einladen dieser Graphikerweiterung läuft.

Wie gesagt, werden Sie zunächst aufgefordert, eine Funktion einzugeben. Das hierzu verwendete INPUT steht als Unterprogramm in Zeile 380 und kann so von Ihnen durch eine andere Eingabeschleife - z.B. mit GET - ersetzt werden. Die Verlegung nach Zeile 380 war notwendig, da Sie bis zur Zeile 330 ja keinerlei Änderungen vornehmen dürfen. Die Funktion steht dann in A$ und wird an die Übersetzerroutine bis Zeile 330 Übergeben. Diese formt daraus dann eine DEF FN - Zeile, d.h. Sie POKEt die notwendigen Bytes direkt in den BASIC-Speicher und verändert so die Zeile 330. Sie können sich ja einmal diese Zeile anschauen, nachdem Sie die erste Funktion eingegeben haben. Die Adresse, bei der die besagte Zeile beginnt, bzw. bei der das Programm beginnt, seine Funktion einzuschreiben, steht in Zeile 230 und kann von Programmierern geändert werden, die die Routine verstanden haben und in diesem Bereich Änderungen vornehmen wollen.

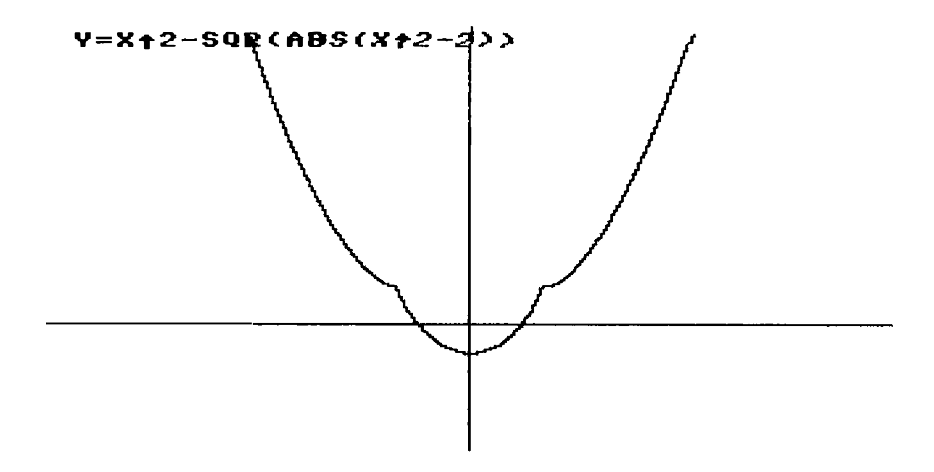
Grundsätzlich können Sie in Ihre Funktion jede der 25 Teilfunktionen aufnehmen, die Sie in den Zeilen 350/360 finden. Dabei ist die Syntax genau dieselbe, wie in BASIC.

Nach den obligatorischen Sprungadressendefinitionen werden Sie aufgefordert, die beiden Verzerrfaktoren f1/f2 und die Verschiebesummanden a und b einzugeben (Z. 490-530). Hier können Sie das anwenden, was Sie in den obigen Ausführungen gelernt haben. Hier einige Richtwerte: f1 und f2 werden bei den meisten Funktionen größer als 1 (bei Winkelfunktionen ist z.B. der Bereich von 10-70 empfehlenswert). Wie Sie wissen ist f1 dafür zuständig, den Graphen nach links und rechts auseinander zu ziehen. f2 dagegen dehnt die Kurve nach oben und unten. Für einen ersten Überblick empfehlen sich für a und b die Werte a=160 und b=100, wodurch der Nullpunkt des Koordinatensystems genau in die Mitte des Bildschirms gebracht wird. Wollen Sie dann z.B. nur die für x positiven Bereiche, dann wählen Sie a=0 usw.

Nach dem Einschalten und eventuellen Löschen der Graphik (abhängig vom Löschflag) werden dann ab Zeile 600 die Achsen des Koordinatensystems gemäß der gewählten Verschiebung gezeichnet. Wenn Sie wollen, können Sie hier noch Teilstriche als Einheiten entlang den Achsen zeichnen.

In Zeile 650 beginnt nun die eigentliche Zeichenarbeit. Sie finden hier die altbekannte Routine vor, die wir oben entwickelt haben. Lediglich eine Kleinigkeit ist verändert. Um nicht nur die einzelnen Punkte der Kurve zu sehen, die berechnet wurden, werden diese hier durch Linien miteinander verbunden. So entsteht ein schönes, angenähertes Bild. Man spricht hier auch von Aproximation. Die Schwierigkeit besteht darin, daß gerade dann nicht vom letzten Punkt zum gerade berechneten gezeichnet werden darf, wenn der Graph aus einem ungültigen Bereich kommt. Dafür sorgt das Flag FL% (Integervariable).

Ist die Kurve gezeichnet, so kommen Sie nach einem Tastendruck in das Hauptmenü. Hier können Sie verschiedene Optionen wählen. Sie können eine ganz neue Funktion wählen, nur die Parameter verändern, das alte Bild erhalten, während das neue gezeichnet wird (für Vergleiche besonders geeignet), Graphik speichern / laden, eine Hardcopy anfertigen oder einfach beenden.



Noch etwas zur Funktioneneingabe: Bitte achten Sie darauf, daß erstens die Syntax Ihrer Funktion richtig ist - andernfalls entsteht ein SYNTAX ERROR. Zweitens sollten Sie darauf achten, daß keine undefinierten Werte eingesetzt werden. Dies sind z.B.: allgemein zu hohe Werte, der Wert 0 in Nennern oder bei Logarithmus, negative Werte bei Wurzel oder Logarithmus.

Negative Werte können z.B. ausgeschlossen werden durch Verwendung der Absolut-Funktion, z.B.:

```
SQR(ABS(X))
```

Der Wert Null wird vermieden durch ersetzen von X durch den Term (X-(X=0)):

```
statt 1/X    : 1/(X-(X=0))     oder
statt LOG(X): LOG(ABS(X-(X=0)))
```

Wird x gleich 0, so wird der Ausdruck X=0 im Commodore-BASIC gleich -1, ansonsten bleibt er 0. Ähnlich können Größer- oder Kleinerzeichen (>,<) verwendet werden.

Ist ein Fehler aufgetaucht, so können Sie das Programm auch mit RUN 330 starten, wobei die Funktion erhalten bleibt.

3.1.2 3-dimensionale Graphik

Kaum eine Computer-Werbeanzeige und zunehmend auch Anzeigen aus anderen Bereichen verzichten auf wunderschöne meist dreidimensionale Graphikbilder, um Ihre Produkte anzupreisen. Kein Wunder, die Zeichnungen strahlen eine enorme Ästhetik aus, durch die der Betrachter die unendlichen Weiten des Raumes mit dem jeweiligen Werbeobjekt assoziiert. Immer mehr Graphikdesigner besinnen sich auf das Hilfsmittel Computer und seine Fähigkeit auch komplizierte 3-dimensionale Funktionen und allgemein räumliche Bilder herzustellen. Wer hier nicht hinterher hinken will, der sollte sich einmal informieren.

Aber auch dem privaten Anwender bereitet es ungeheuren Spaß, sich mit diesen Dingen zu beschäftigen. Nicht umsonst ist meist eines der ersten Objekte des Informatikunterrichts in Schulen die Erzeugung von einfachen zweidimensionalen Funktionsgraphen (s. vorherigen Abschnitt) bis hin zu den räumlichen Darstellungen.

Doch stößt Letzteres ohne Vorkenntnisse und ohne sachkundige Anleitung auf erhebliche Schwierigkeiten, da die Fachliteratur oft reichhaltiges Wissen voraussetzt und die eigene Ableitung der Algorithmen (Rechenvorschriften) relativ kompliziert ist. Aus diesem Grunde seien hier einige Verfahren dargelegt, die sich auf die Kenntnis des in Abschnitt 3.1.1 Gesagten stützen.

3.1.2.1 Parallel-Projektion

Man unterscheidet grundsätzlich zwei Arten von Projektionen:

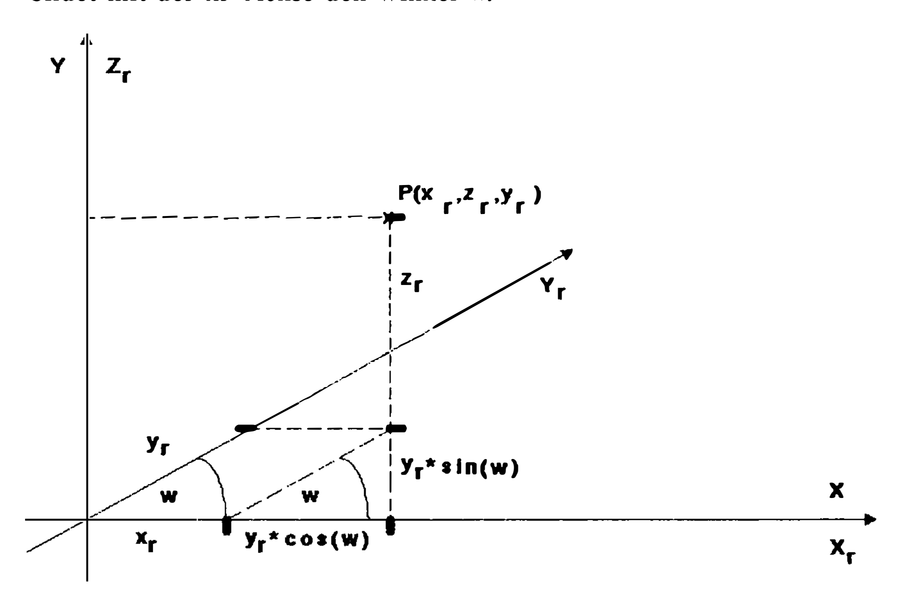
- parallele Projektion
- Zentralprojektion

Wir werden uns im folgenden mit der ersten einfachen Projektionsart beschäftigen. Weiter unten liefern wir auch das nötige Rüstzeug für die zweite, wirklichkeitsnahere.

Zunächst einmal müssen wir unser 2-dimensionales Koordinatensystem erweitern. Hierzu verwenden wir die Achsen xr,yr und zr (r steht für räumlich). Jeder Punkt eines 3-D-Objektes (z.B. ein Haus) hat damit drei Koordinaten (xr,yr,zr). Dieses Koordinatentripel muß zur Darstellung auf dem Bildschirm (denn hier kennen wir ja nur die Koordinaten x und y) in die ebenen Koordinaten x und y umgerechnet werden.

Bevor wir dies unternehmen, müssen wir uns erst einmal über das Aussehen unseres 3-D-Koordinatensystems einigen. Wir legen dafür hiermit folgendes fest (s.Abbildung):

Die xr- und zr-Achsen liegen in der Zeichenebene und stehen senkrecht aufeinander. Die durch die Zeichenebene gehende yr-Achse bildet mit der xr-Achse den Winkel w:



Wie aus der Zeichnung ersichtlich, ergeben sich damit folgende Formeln, die die Umrechnung der Raum- auf die Ebenenkoordianten vornehmen:

```
x = xr + yr*cos(w)
y = zr + yr*sin(w)
```

Der Winkel w bestimmt dabei die Perspektive, also den Winkel, mit dem man das Objekt betrachtet. Wollen wir das Bild verschieben, so daß wir es z.B. mitten im Graphikfenster zu sehen bekommen oder das Objekt in weite Ferne zu rücken, können wir drei Summanden a,b und c zu der xr-,zr- und yr-Koordinate hinzuaddieren (s. Kap. 3.1). Wir erhalten dann:

```
x = a+xr + (yr+c)*cos(w)
y = b+zr + (yr+c)*sin(w)
```

Wollen Sie die Figur lediglich in der Ebene verschieben, so wählen Sie c=0. Doch auch hier wird es oft dazu kommen, daß Teile des Bildes nicht zu sehen sind, da sie entweder aus dem Bildschirm herausragen oder viel zu klein sind, als daß die Auflösung ausreichte, sie zu erkennen. Aus diesem Grunde führen wir wieder unsere bekannten Verzerrfaktoren ein, die uns erlauben sollen, das Objekt einfach zu vergrößern oder Länge, Höhe und Breite zu verändern. Um alle drei Faktoren zu verändern, müssen wir wieder direkt die Raumkoordinaten mit den drei Faktoren f1,f2 und f3 für die xr-, zr- und yr-Koordinaten:

```
x = f1*(a+xr) + f3*(yr+c)*cos(w)
y = f2*(b+zr) + f3*(yr+c)*sin(w)
```

Wollen Sie die Figur nur vergrößern, ohne Sie zu verzerren, so wählen Sie f1=f2=f3. Für die drei Faktoren gilt wieder das in Abschnitt 3.1 Gesagte:

	f1/2/3 > 1 =>	Streckung (Vergrößerung)
0 <	f1/2/3 < 1 =>	Stauchung (Verkleinerung)

Um die jeweilige Figur nun noch genau auf den Bildschirm zu bekommen, da der Nullpunkt der Raumkoordinaten nicht unbedingt immer mit dem Nullpunkt des Bildschirms übereinstimmen soll, der bekanntlich links oben in der Ecke liegt, wollen wir noch zwei Verschiebesummanden v1 und v2 einführen. v1 verschiebt das planare

Koordinatensystem in plus-x-, v2 in minus-y-Richtung. Wollen Sie den Nullpunkt des Koordinatensystems beispielsweise genau in die Mitte des Bildschirms positionieren, so wählen Sie v1=160 und v2=100. Auch hier stehen unsere y-Ebenen-Koordinaten wieder auf dem Kopf, wir müssen also ihr Vorzeichen ändern. Damit ergeben sich die beiden folgenden Formeln:

```
x =  f1*(a+xr) + f3*(yr+c)*cos(w) + v1
y = -f2*(b+zr) - f3*(yr+c)*sin(w) + v2
```

In dem folgenden Programm wird ein Haus räumlich dargestellt. Die dazu notwendigen Eckpunkte sind mit Ihren 3 räumlichen Koordinaten in DATA-Zeilen niedergelegt (Zeilen 1110-1140). Die 3-D-Koordinaten werden in die der Ebene umgerechnet und die einzelnen Punkte miteinander verbunden. Welche Punkte jeweils verbunden werden sollen, steht ebenfalls in dahinter liegenden DATA-Zeilen. Die Punkte werden der Reihe nach von 1 ab durchnumeriert und jeweils die Nummern zweier zu verbindender Punkte hintereinander in den Linien-DATA-Zeilen aufgeführt. Die Reihenfolge, in der die Linien gezeichnet werden, ist völlig egal, sie brauchen auch nicht zusammen zu hängen. Jeweils zu Anfang der Punkt- und der Linien-Daten steht die Anzahl der Punkte (Zeile 1100) und der Linien (Zeile 2100), die gezeichnet werden sollen. Wenn Sie sich an diese Datenform halten, können Sie sich beliebige eigene Figuren ausdenken und räumlich darstellen. Beispielsweise können Sie doch einmal Ihren Computer vermessen und die Daten in das Programm eingeben. Sie erhalten dann ein schönes räumliches Bild Ihres Gerätes.

Doch wofür sich die Umstände machen und die Koordinaten 3-dimensional eingeben? Dies hat einen besonderen Grund. Sie können Ihre einmal so erstellte Figur, wie in unserem Fall das Haus, beliebig vergrößern, verzerren, perspektivisch drehen oder verschieben. Dies erreichen Sie durch Veränderung der einzelnen Parameter a,b,c, f1,f2,f3, v1,v2 und w. Es wird eine Weile dauern, bis Sie die Auswirkungen dieser vielen verschiedenen Variablen überblicken. Bedenken Sie, daß die Zerrfaktoren f1,f2 und f3 die Einheiten der Achsen bestimmen, d.h. bei ihrer Veränderung scheinen sich die Objekte gleichfalls fort oder heran zu bewegen. Stört Sie das, so ändern Sie die einfach die obigen Formeln in jene:

```
x =  f1*a+xr + f3*yr*cos(w)+c + v1
y = -f2*b-zr - f3*yr*sin(w)-c + v2
```

Nun besitzen f1, f2 und f3 keinen Einfluß mehr auf den Abstand, obwohl diese Formeln mathematisch nicht ganz einwandfrei sind. Die Verzerrung bezieht sich nur auf das Objekt, nicht auf das Koordinatensystem. Wollten Sie also Einheiten an den Achsen abtragen, käme es zu Unstimmigkeiten.

Eine sicher interessante Sache ist in den Zeilen 600-620 niedergelegt. Wir haben an dieser Stelle gerade die Raum-x- und die Raum-z-Achse gezeichnet. Um nun auch die in die Zeichenebene hineinragende y-Achse zu zeichnen, berechnen wir einfach den Punkt, bei dem die y-Ebenen-Koordinate 0 (bzw. 199) ist (unter Berücksichtigung des Winkels w) und zeichnen eine Linie vom Ursprung dorthin.

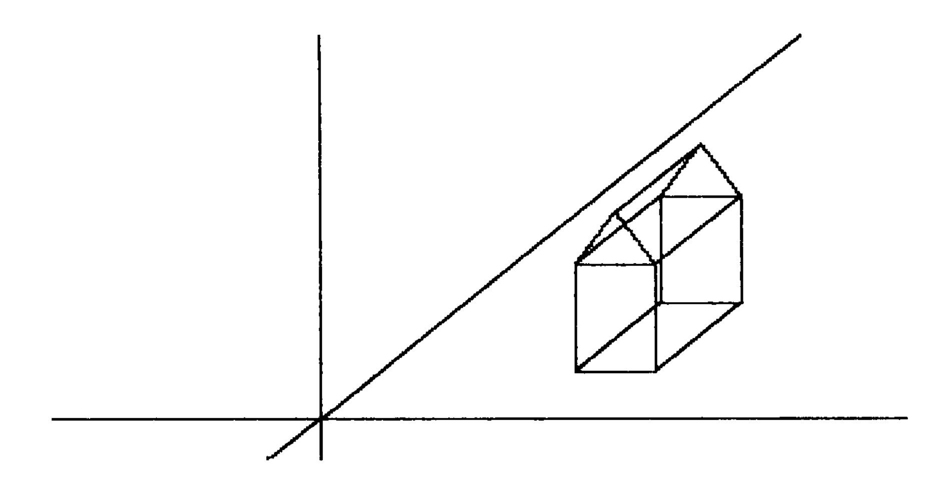
Wichtig bei allen Zeichenvorgängen ist die vorherige Überprüfung der Werte auf Bereichsüberschreitungen. Ein sehr gutes, professionelles Zeichenprogramm würde, falls ein oder zwei Eckpunkte außerhalb des Bildschirms liegen, trotzdem aber noch Teile der Linie zu sehen wären, die Schnittpunkte der Linie mit den Fensterränder berechnen und den sichtbaren Teil der Linie zeichnen. Unser einfaches Programm zeichnet diese Linien gar nicht. Sie könnten da ja Abhilfe schaffen.

Gleichfalls werden die Linien, die eigentlich verdeckt sein sollten, ebenfalls gezeichnet. Dieses Problem der verdeckten Linien gehört zu den schwierigsten und langwierigsten Kapiteln der 3-D-Graphik. Große Institute tüfteln mit teilweise großem mathematischen Aufwand an solchen Algorithmen. Weiter unten wird Ihnen ein äußerst einfacher vorgestellt, der insbesondere bei 3-dimensionalen Funktionen Verwendung findet.

Vielleicht verändern Sie dieses Programm noch weiter. Sie könnten z.B. die jetzt in DATA-Zeilen abgelegten Daten in irgendeiner Form auf Diskette abspeichern und gegebenenfalls später wieder einladen (DATA-Zeilen wären hierfür natürlich ungeeignet. Sie könnten alle Daten in Arrays unterbringen, die als sequentielles File auf Diskette gebracht werden können). Oder Sie bauen das Programm zu einem richtigen menügesteuerten Zeichenprogramm aus; als kleines Beispiel dient Ihnen dazu schon das Programm aus Paragraph 3.1.

```
100 REM    *********************
110 REM    **                 **
120 REM    **  3-D-DESIGNER   **
130 REM    **                 **
140 REM    *********************
150 REM
400 REM
410 REM         *********
420 REM   ****  PARAMETER  ***
430 REM         *********
440 REM
450 F1 = 5 : F2 = 5 : F3 = 3 : REM ZERRUNG
460 AR = 15 : BR =  0 : CR = 10 : REM VERSCHIEBUNG DER RAUMKOORD.
470 W =  /4: REM SICHTWINKEL (IN RAD)
480 SI = SIN(W) : CO = COS(W) : REM KONSTANTEN
490 V1 = 100 : V2 = 180 : REM VERSCHIEBUNG DER EBENENKOORD.
500 REM
510 REM         ***********
520 REM   ****  UMRECHNUNG  ***
530 REM         ***********
540 REM
550 GMODE 0,1 : GCLEAR : SCOL= 0,16: SCOL=1,8 : REM GRAPHIK INIT
560 FG=0:IF V1<0 OR V1>319 THEN FG=1:GOTO 580 : REM FLAG
570 PLOT ,V1,0 TO V1,199 : REM RAUM-Z-ACHSE
580 IF V2<0 OR V2>200 THEN FG=1:GOTO 600
590 PLOT ,0,V2 TO 319,V2 : REM RAUM-X-ACHSEE
600 Z2 = V1 - (199-V2)/SI*CO : Z1 = V1 + V2/SI*CO
610 IF FG=0 AND Z1>=0 AND Z1<320 THEN:PLOT ,V1,V2 TO Z1,0: REM RAUM-Y-
ACHSE OBEN
620 IF FG=0 AND Z2>=0 AND Z2<320 THEN:PLOT,V1,V2 TO Z2,199: REM RAUM-Y-
ACHSE UNTEN
630 READ AP : DIM X%(AP),Y%(AP) : REM ANZAHL PUNKTE
640 FOR ZA=1 TO AP : REM PUNKTEZAHL
650 READ XR,ZR,YR
660 X%(ZA) =  F1*(XR+AR) + F3*(YR+CR)*CO+V1
670 Y%(ZA) = -F2*(ZR+BR) - F3*(YR+CR)*SI+V2
680 NEXT ZA : REM NAECHSTER PUNKT
700 REM
710 REM         *******
720 REM   ****  STRICHE  ***
730 REM         *******
740 REM
750 READ AL : REM ANZAHL LINIEN
760 FOR ZA=1 TO AL
770 READ P1,P2 : REM PUNKTNUMMERN LESEN
775 IF X%(P1)<0 OR Y%(P1)<0 OR X%(P2)<0 OR Y%(P2)<0 THEN GOTO 790:REM
AUSSERHALB
777 IFX%(P1)>319ORY%(P1)>199ORX%(P2)>319ORY%(P2)>199THEN GOTO 790:REM
AUSSERHALB
780 PLOT ,X%(P1),Y%(P1) TO  X%(P2),Y%(P2) : REM VERBINDEN
790 NEXT ZA : REM NAECHSTE LINIE
```

```
900 POKE 198,0 : WAIT 198,255 : GET A$
910 GMODE ,0 : LIST : REM GRAPHIK AUS
1000 REM
1010 REM      ************
1020 REM ****  KOORDINATEN  ****
1030 REM      ************
1100 DATA 10 : REM ANZAHL PUNKTE
1110 DATA  0, 0, 0,  6, 0, 0,  6,10, 0
1120 DATA  0,10, 0,  3,15, 0,  3,15,15
1130 DATA  6,10,15,  6, 0,15,  0, 0,15
1140 DATA  0,10,15
2000 REM
2010 REM      *************
2020 REM ****  VERBINDUNGEN  ****
2030 REM      *************
2100 DATA 17 : REM ANZAHL LINIEN
2110 DATA  1, 2,  2, 3,  3, 4,  4, 1
2120 DATA  4, 5,  5, 3,  5, 6,  6, 7
2130 DATA  7, 3,  7, 8,  8, 2,  8, 9
2140 DATA  9, 1,  9,10, 10, 4, 10, 6
2150 DATA 10, 7
```



Wenn Sie dieses Programm eingeladen und ausprobiert haben, werden Sie sehen, welchen Spaß es macht, mit ihm zu arbeiten. Speichern Sie ruhig einmal gelungene Bilder ab, oder erstellen Sie eine Hardcopy auf Ihrem Drucker.

Wollen wir die einzelnen Objekte zusätzlich im Raum drehen (nicht nur die Perspektive ändern), so müssen wir aus den ungedrehten Raumkoordinaten xr,yr,zr erst die gedrehten xr',yr',zr' errechnen, bevor wir sie in Ebenenkoordinaten umrechnen können. Dies geschieht mit Hilfe der folgenden Formeln:

Drehung um die xr-Achse:

```
xr' =  xr
yr' =  yr*cos(u) + zr*sin(u)
zr' = -yr*sin(u) + zr*cos(u)
```

Drehung um die yr-Achse:

```
xr' =  xr*cos(t) + zr*sin(t)
yr' =  yr
zr' = -xr*sin(t) + zr*cos(t)
```

Drehung um die zr-Achse:

```
xr' =  xr*cos(s) + yr*sin(s)
yr' = -xr*sin(s) + yr*cos(s)
zr' =  zr
```

Dabei bedeuten:

u,s,t : Drehwinkel um die jeweiligen Achsen

Wollen Sie um zwei Achsen drehen, dann müssen diese Drehungen nacheinander ausgeführt werden: Sie berechnen erst die gedrehten Koordinaten für die Drehung um die erste Achse, dann setzen Sie die erhaltenen Werte in die Gleichungen ein, die für die Drehung um die zweite Achse zuständig sind. Entsprechendes gilt bei der Drehung um alle Achsen.

Die Erstellung dreidimensionaler Bilder findet in vielen Bereichen Anwendung. Da mit dem Computer auf die besprochene Art und Weise besonders gut reale Gegenstände oder Situationen maßstabsgetreu dargestellt und verändert werden können, eignen sich diese Techniken für die Konstruktion bzw. den Entwurf bestimmter Objekte wie

Gebäude, Einrichtungen, Maschinen oder Maschinenteile. Dieses computerunterstützte Entwerfen (auch CAD = Computer Aided Design genannt) ist in vielen Teilen der Indrustrie und Ingenieurtätigkeit zu einem nicht mehr weg zu denkenden Werkzeug geworden. Autos, Flugzeuge werden konstruiert, technische Zeichnungen angefertigt oder Motoren etc. auf den Bildschirm und dann auf Papier gebracht. In diesem Zusammenhang eignet sich die 3-dimensionale Graphik hervorragend zur Simulation auch komplizierter Vorgänge. Diese Möglichkeit des Computers ist schon relativ früh entdeckt worden, zu einer Zeit, in der ein einfacher Taschenrechner noch ein Vermögen kostete, kompliziertere Geräte aber Unsummen verschlangen, was wohl ein Zeichen für die Wichtigkeit solcher Techniken ist.

3.1.2.2 Zentral-Projektion

Wenn wir Gegenstände betrachten, so können wir Ihre Entfernung unter anderem dadurch feststellen, daß sie je entfernter sie stehen desto kleiner werden. Das klassische Beispiel hierzu sind die langen Eisenbahnschienen, die immer schmaler werden und sich weit weg mit dem Horizont treffen. Tatsächlich ist es so, daß alle Linien sich auf einen virtuellen Punkt zuzubewegen scheinen, den sogenannten Fluchtpunkt. Das Problem ist es nun, ein mathematisches Verfahren zu entwickeln, um unsere bisher parallelen Linien auf einen Punkt zu richten. Da die Ableitung der entsprechenden Formeln hier zu lange dauern würde, seien Sie hier mit Scalierung (f1/f2/f3) und Verschiebung (a/b/c) angegeben. Dabei stehen nun alle Achsen aufeinander senkrecht und die z-Achse ragt in die Bildebene hinein (eben war es die y-Achse!):

```
x = (f1*xr+a)/q
y = (f2*yr+b)/q
mit: q = 1-(f3*zr+c)/fpz
```

dabei bedeuten nun:

xr,yr,zr: räumliche Koordinaten
f1,f2,f3: Verzerrung in xr-,yr-,zr-Richtung
a, b, c: Verschiebung in xr-,yr-,zr-Richtung
fpz: Entfernung (z-Koord.) des Fluchtpunktes

Sie müssen also erst q errechnen, um die beiden gesuchten Ebenen-Koordinaten zu errechnen. Wollen Sie gleichzeitig Ihr Objekt noch in alle drei Richtungen drehen, so wird die Sache sehr viel komplizierter:

```
x = (f1*(A*xr + D*yr + G*zr)+a) / q
y = (f2*(B*xr + E*yr + H*zr)+b) / q
mit:  q = 1-(f3*(C*xr + F*yr + I*zr)+c) / fpz
```

Das sieht schon kompliziert genug aus. Wenn Sie aber sehen, was Sie für A,B,C,...I einsetzen sollen, werden Ihnen die Augen ausgehen:

```
A =  cos(s)*cos(t)
B =  sin(s)*cos(t)
C = -sin(t)
D = -sin(s)*cos(u) + cos(s)*sin(t)*sin(u)
E =  cos(s)*cos(u) + sin(s)*sin(t)*sin(u)
F =  cos(t)*sin(u)
G =  sin(s)*sin(u) + cos(s)*sin(t)*cos(u)
H = -cos(s)*sin(u) + sin(s)*sin(t)*cos(u)
I =  cos(t)*cos(u)
```

Dabei bedeuten:

s: Winkel der Rotation um die z-Achse
t: Winkel der Rotation um die y-Achse
u: Winkel der Rotation um die x-Achse

Solche Mammutaufgaben können selbstverständlich nur in Maschinensprache in relativ angemessener Zeit bewältigt werden. Selbst da gibt es einige Zeitprobleme (ein Tip am Rande: Die beste Möglichkeit wäre das Anlegen einer Tabelle von Sinus- und Cosinuswerten (jeweils in einem bestimmten Winkelabstand (z.B.1 Grad etc.)), in der das Programm bei Bedarf immer nachschlagen kann, ohne die Werte erst lange errechnen zu müssen).

3.1.2.3 3-D-Funktionen

Eine sehr reizvolle Anwendung der 3-D-Technik ist die Darstellung dreidimensionaler Funktionen. Doch nicht nur dem bloßen Vergnügen des Erstellers oder einiger Mathematiker dient diese weitere Möglichkeit. Auf diese Weise sind komliziertere Zusammenhänge, die sonst aus vielen unübersichtlichen Tabellen erahnt werden müssen, äußerst plastisch und informativ darstellbar. Sie kennen den Nutzen von zweidimensionalen Diagrammen, die beispielsweise die Entwicklung des jährlichen Umsatzes im Laufe der Jahre wiederspiegeln. Wollte man nun etwa gleichzeitig noch die Abhängigkeit des Umsatzes vom Preis eines bestimmten Produktes in den verschiedenen Jahren darstellen, so müßte man für jedes Jahr ein solches Diagramm erstellen, was auf die Dauer zu einer unübersehbaren Schar von Kurven führte. Doch dieser komlexe Zusammenhang kann anschaulich in einem einzigen 3-D-Diagramm vermittelt werden. So erhalten Sie einen schnellen und guten Überblick über die Dinge und können so gemäß den Trends den optimalen Preis Ihres Produktes im nächsten Jahr (Monat) ermitteln. Ich brauche Ihnen nicht erst zu sagen, welche Vorteile Ihnen dadurch erwachsen.

Hier soll Ihnen die Technik der Erstellung anhand von mathematischen Funktionen vermittelt werden. In diesem Fall werden die einzelnen Werte, die notwendig sind, um das Bild zu erstellen, errechnet. Sie können diese selbstverständlich auch aus irgendwelchen Tabellen ermitteln, was den Anwendungsbereich der jetzt beschriebenen Vorgänge enorm erweitert.

Zu dem Graphen einer 3-D-Funktion kommen wir durch Verwendung einer sogenannten Wertematrix. Als Beispiel soll uns die Funktion

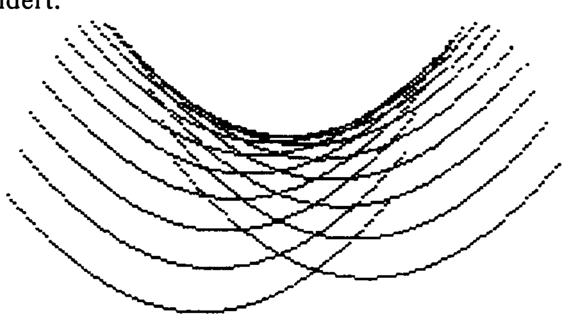
```
z = x^2 + y^2
```

dienen. Sie erinnern sich aus Abschnitt 3.1.1, daß dort die Funktionswerte f(x) in einer FOR...NEXT-Schleife ermittelt wurden, in der die Variable x (die sogenannte Laufvariable) stets aufgezählt und die Variable y errechnet wurde. So erhielten wir beliebig viele Wertepaare, die uns als Koordinaten für den Graphen unserer Funktion dienten.

Bei dreidimensionalen Funktionen (f(x,y)) dagegen besitzen wir zwei Laufvariablen (x und y), mit denen wir die dritte (z) berechnen müssen. Um diesem Umstand gerecht zu werden, verwenden wir zwei ineinander geschachtelte FOR...NEXT-Schleifen. In der ersten wird der x-Wert, in der zweiten der y-Wert hochgezählt. Das folgende Programm bringt Sie diesem Sachverhalt näher:

```
100 REM ***********************
110 REM **                   **
120 REM **  3-D: Z=Y^2-X^2   **
130 REM **                   **
140 REM ***********************
150 REM
160 REM
170 REM **** PARAMETER ****
180 REM      *********
190 W = 3.1415/8
200 A  =   0 : B  =   0 : C  = 0
210 F1 =  20 : F2 =   5 : F3 = 8
220 V1 = 160 : V2 = 100
230 CO = COS(W) : SI = SIN(W)
240 REM
250 REM **** ZEICHNEN ****
260 REM      ********
270 GMODE 0,1 : GCLEAR : SCOL= 0,16 : SCOL= 1,4 :REM GRAPHIK INIT
280 FOR YR=3 TO -4 STEP -0.5
290 FOR XR=3 TO -3 STEP -0.05
300 ZR=YR*YR-XR*XR          :REM FUNKTION
310 X=F1*(A+XR) + F3*(YR+C)*CO + V1
320 Y=F2*(B+ZR) + F3*(YR+C)*SI + V2
330 PLOT ,X,Y               :REM PUNKT ZEICHNEN
340 NEXT XR,YR
350 POKE 198,0 : WAIT 198,255
```

Die besagten FOR...NEXT-Schleifen überstrecken, wie Sie vielleicht sehen, die Zeilen 280 bis 340. In 300 wird dann aus den beiden Laufvariablen der zr-Wert berechnet, um dann diese Raumkoordinaten auf altbekannte Weise in Ebenenkoordinaten umzurechnen. Das einzige Problem ist hier, wie immer, die Wahl der Parameter und des xr-, yr-Wertebereichs. Bei 3-D-Funktionen tritt es besonders auf, da hier so viele Parameter zu bestimmen sind. Man hilft sich, indem man zunächst möglichst einfache Zahlen einsetzt (etwa: a,b,c=0, f1,2,3=1, w=3.1415/4, v1=160, v2=100) und dann das entstehende Bild entsprechend verändert.



Vernetzung:

Durch einen kleinen Trick können wir nun einen ganz besonders schönen Effekt mit hineinbringen. Bislang erhalten wir nur einzelne Kurven, die uns schrittweise den Verlauf in die dritte Dimension vermitteln. Oft sieht man aber quasi gebogene Gitternetze, die uns die Graphik noch ein ganzes Stück plastischer erscheinen lassen (Vernetzung oder "crosshatching"). Dies wird einfach durch eine scheinbare Drehung der Kurve erreicht. Scheinbar deshalb, weil es sich eigentlich gar nicht um eine Drehung handelt, sondern nur eine Veränderung der Schrittweiten, in denen die xr-, bzw. yr-Werte vom Start- zum Endwert gelangen, also derjenigen Werte, die hinter den STEP-Komandos der beiden FOR...NEXT-Schleifen erscheinen.

Im obigen Beispiel wurde yr mit einer Schrittweite von -0.5, xr dagegen mit -0.05 aufgezählt. Dadurch erschien nicht etwa eine gleichmäßige Fläche, was passieren würde, wenn wir beide Schrittweiten gleich groß wählten, sondern jenes bekannte "Streifenmuster".

Wenn wir jetzt die Schrittweiten austauschen, so verläuft unser Streifenmuster genau senkrecht zum ersten. Das wird im nächsten Programm ausgenutzt:

```
100 REM **********************
110 REM **                  **
120 REM **  3-D: Z=Y^2-X^2  **
130 REM **    VERNETZUNG    **
140 REM **********************
150 REM
160 REM
170 REM **** PARAMETER ****
180 REM      *********
190 W = 3.1415/8
200 A  =   0 : B  =   0 : C = 0
210 F1 =  20 : F2 =   5 : F3 = 8
220 V1 = 160 : V2 = 100
230 CO = COS(W) : SI = SIN(W)
240 REM
250 REM **** ZEICHNEN ****
260 REM      ********
270 GMODE 0,1 : GCLEAR : SCOL= 0,16: SCOL= 1,4 :REM GRAPHIK INIT
280 SY=-0.5 : SX=-0.05    :REM SCHRITTWEITEN (STEPS)
290 FOR ZA=1 TO 2         :REM ZAEHLER
300 FOR YR=3 TO -4 STEP SY
```

```
310 FOR XR=3 TO -3 STEP SX
320 ZR=YR*YR-XR*XR          :REM FUNKTION
330 X=F1*(A+XR) + F3*(YR+C)*CO + V1
340 Y=F2*(B+ZR) + F3*(YR+C)*SI + V2
350 PLOT ,X,Y               :REM PUNKT ZEICHNEN
360 NEXT XR,YR
370 IF ZA=1 THEN GOSUB 410: SY=-0.05 : SX=-0.5
380 NEXT ZA
390 GCOMB 1 :REM GRAFIKSEITEN ODER-VERKNUEPFEN
400 POKE 198,0 : WAIT 198,255 : GMODE ,0 : END :REM GRAPHIK AUS
410 GMODE 2 : COLOR=0,0 : GCLEAR :REM SEITE 2 EINSCHALTEN UND LOESCHEN
420 RETURN
```

Sie sehen, der Graph wird insgesamt zweimal auf zwei verschiedene Weisen gezeichnet. Beim ersten Mal wird die Zeichnung in der ersten Graphikseite, beim zweiten Mal in der zweiten gezeichnet (dabei zeichnen wir mit vertauschten STEP-Parametern). Zu guter Letzt werden die beiden Graphiken miteinander durch OR verknüpft, was zu einer Überlagerung führt.

In unserem speziellen Beispiel wäre das getrennte Zeichnen in zwei Graphikseiten und das nachherige ORen nicht nötig, wir könnten also direkt das zweite Bild über das erste zeichnen. Doch für unseren nächsten Schritt ist dieses Verfahren unabdingbar.

Vielleicht bringen Sie noch ein 3-D-Koordinatensystem in das Bild, wie es Ihnen das vorherige Programm zeigt.

Versteckte Linien:

Im diesem Teil werden wir ein einfaches Verfahren für das Auslöschen eigentlich von vorderen Flächen verdeckter Linien beim Zeichnen mathematischer und auch anderer Funktionen vorstellen.

Im täglichen Leben können wir normalerweise nicht durch die Dinge schauen, die wir betrachten. Wie Sie jedoch beim Zeichnen unserer Funktion sehen, ist dies hier der Fall. Unsere Linien durchdringen sich, obwohl die vorderen, die ja eine Fläche abstecken, die hinteren verdecken müßten.

Bei Funktionen gibt es nun unter anderem zwei Methoden, diese verdeckten Linien zu unterdrücken bzw. nachträglich wieder zu löschen. Fangen wir bei der ersten an. Sie ist äußerst simpel, aber sehr effektiv und trickreich:

Beim Zeichnen unseres Graphen achten wir darauf, daß wir stets von hinten nach vorne, also mit abnehmendem yr zeichnen. Haben wir jetzt einen Punkt ganz normal ausgerechnet und auf den Bildschirm gesetzt, dann löschen wir einfach unter dem Punkt in einer Linie alles bis zum unteren Ende des Graphikfensters. Damit verdeckt jede neu gezeichnete Ebene alles hintere, vorausgesetzt, es liegt räumlich unter dem gezeichneten Punkt. So erhalten wir eine Aufsicht auf die graphische Struktur der Funktion. Wenn Sie in das obige Programm lediglich eine einzige Zeile hinzufügen, wird Ihnen der Effekt deutlich:

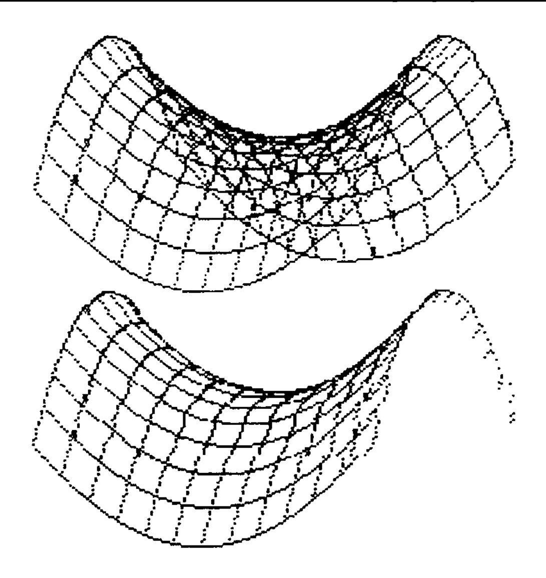
```
355 PLOT 1,X,Y+1 TO X,199 :REM UNTER PUNKT LOESCHEN
```

Das einzige Problem sind eventuell die Randbereiche. Hier könnte man eigentlich die Sicht quasi von unten auf die Kurve zulassen. Doch mit unserem Algorithmus ist dies nicht möglich - wie Sie sehen, wenn Sie das Programm laufen lassen -, da auch die "Rückansicht" gelöscht wird. Um dieses Manko auszugleichen, könnte man nun nicht bis zum unteren Bildrand, sondern lediglich zum darunterliegenden Punkt der nächsten Linie löschen. Damit wäre das Problem erledigt. Das Zeichnen würde mit dieser Methode jedoch um Einiges länger dauern. Vielleicht versuchen Sie einmal das obige Programm so umzuschreiben. Es gibt einige schöne Funktionen, die Sie ausprobieren sollten. Versuchen Sie es evt. 'mal mit jenen:

```
z = x^2+y^2
z = 1/(1+x^2+y^2)
z = SQR(1-x^2/4-y^2/9)
z = sin(x)/x+sin(y)/y
z = sin(1/x)/x+sin(1/y)/y
```

Sie können diese Funktionen auch mischen, da der Teil, der x enthält den Verlauf in x-Richtung wiederspiegelt, der Teil mit y den in y-Richtung. Man käme dann beispielsweise auf so etwas (Mischung: letzte und vorletzte):

```
z = sin(x)/x+sin(1/y)/y
```



3.1.2.4 Bewegte Bilder in 3-D

In den letzten Paragraphen haben Sie die Methoden kennengelernt, die zur Erzeugung 3-dimensionaler Bilder notwendig sind. Wenn es nicht gerade kleine Objekte waren, dauerte die Erstellung recht lange. Wie sollten daraus laufende Vorgänge entstehen, die von uns als bewegt angesehen werden können? Nun es gibt im Prinzip hierbei zwei Möglichkeiten. Die erste ist relativ einfach: Sie erstellen ein Bild unsichtbar in einem im Moment nicht angezeigten Graphikbereich während Sie den Inhalt eines anderen Graphikbereiches auf dem Bildschirm darstellen. Ist das Bild vollendet, so schalten Sie einfach auf die neue Graphikseite um (s. Kap. 4.3.2). Die nun unsichtbar gewordene erste Seite kann dann zur Erstellung des nächsten Bildes herangezogen werden usw.

Zwar ereichen Sie damit keine besonders schnell bewegten Bilder, doch ist für uns trotzdem ein gewisser Bewegungseffekt vorhanden. Sehr schön sind Rotationen oder schrittweise Vergrößerungen der Objekte.

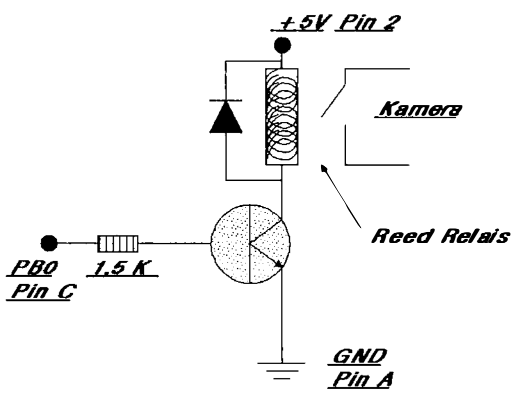
Die zweite Methode ist ein wenig aufwendiger. Sie benötigen dafür eine Filmkamera, die mit Einzelbildschaltung und einem Fernauslöser ausgerüstet ist. Rein theoretisch ist es auch möglich, eine Kamera ohne eine solche Einrichtung zu verwenden. Sie müssen dann zum Filmen einer Situation extrem kurz auf den Auslöser drücken, um möglichst wenige Bilder auf einmal zu knipsen. Mit etwas Übung erhalten Sie dann ebenfalls recht schöne Ergebnisse. Sind diese Voraus-

setzungen geschaffen, dann können Sie Ihre eigenen Computergraphikfilme produzieren. Was sonst nur im Kino oder Fernsehen in Science fiction-Filmen zu sehen ist, das bekommen Sie nun hautnah ins Haus.

Sie brauchen Ihre Kamera lediglich auf den Fernseher zu fixieren (am besten mit Stativ), die Einzelbildschaltung einzustellen und loszulegen: Sie programmieren dem Rechner eine Folge von Bildern ein. Die Rechen- bzw. Erstellungszeit ist dabei egal. Beispielsweise drehen Sie Ihr Objekt (Kurven oder Gegenstände) mit jedem Male ein Stück um eine Achse. Jedesmal, wenn ein Bild fertig gezeichnet ist, dann drücken Sie zum Festhalten eines Bildes einmal auf den Auslöser. Dies muß in sehr kleinen Schritten erfolgen, damit der Vorgang nachher auf dem Film nicht zu schnell erfolgt. Im Zweifelsfall photographieren Sie jede Szene mehrmals.

Eine noch elegantere Methode ist die Steuerung der Kamera durch den Computer. Dies ist dann möglich, wenn Ihre Kamera einen elektrischen Fernauslöser besitzt. Haben Sie nur einen Drahtauslöser, dann gibt es entsprechende Adapter, die im Fachgeschäft erhältlich sind. Sie können dann Ihren Computer alleine laufen lassen, ohne selbst jedesmal die Kamera betätigen zu müssen. Gleichfalls können Sie, wenn Sie keine Einzelbildschaltung besitzen, extrem kurze Schaltzeiten erreichen und so nur wenige oder gar nur ein Bild auf einmal filmen.

Zur Verwirklichung dieses Vorhabens müssen wir einen kleinen Abstecher in die Elektronik machen. Der Computer sendet jedesmal, wenn ein Bild geknipst werden soll ein Signal an den User-Port. Mittels der unten angegebenen Schaltung wird dann die Kamera ausgelöst. Sie können sich diese folgende Schaltung selbst oder von kundigen Bekannten zusammenlöten lassen. Weiterhin benötigen Sie einen User-Port-Stecker und einen entsprechenden Stecker für den Fernauslöser Ihrer Kamera, die beide im Elektronik-Fachgeschäft erhältlich sind.



Diese Schaltung wurde eigenhändig ausprobiert und lieferte hervorragende Filmergebnisse. Mit den unten stehenden Befehlsfolgen können Sie die Kamera ansteuern:

```
10 C2=56576 : REM BASISADRESSE CIA 2 ($DD00)
20 POKE C2+3, PEEK(C2+2) OR 1 : REM PIN AUF AUSGANG
30 POKE C2+1,0 : REM KAMERA AUS
40 POKE C2+1,1 : REM KAMERA EIN
```

Zeile 20 braucht nur einmal im Verlaufe des Programms gegeben werden. Beachten Sie bitte, daß zwischen einem 'ein' und 'aus' genügend Zeit ist, daß das Relais und die Kamera reagieren können.

3.1.3 Graphische Statistik

Ein beliebtes Anwendungsgebiet der Graphik, besonders für den kommerziellen Gebrauch, stellt die Veranschaulichung komplizierter Tabellen in übersichtlichen Diagrammen dar. Hierzu werden verschiedene Darstellungsmethoden verwendet. Die wichtigsten unter ihnen sind:

- Kurvenstatistik
- Balkendiagramme
- Kuchendiagramme

a) Kurvenstatistik:

Besitzen wir viele "Meßwerte" in Abhängigkeit eines bestimmten Faktors (z.B. der Umsatz einer Firma in Abhängigkeit von dem jeweiligen Monat (d.h. in jeweils verschiedenen Monaten)), so verwenden wir allgemein die erste Form der Ausgabe. Hierbei werden die gleichen Techniken angewandt, wie dies bei der Darstellung von 2-dimensionalen Funktionen (s. Kap. 3.1.1) der Fall ist. Der Unterschied liegt lediglich in der Herkunft der einzelnen Daten. Dort wurden sie errechnet, hier aus zweireihigen Tabellen gewonnen. Diesen Abschnitt sollten Sie sich also durchgelesen haben. Grundsätzlich gelten dieselben Regeln zur Verschiebung, Scalierung und Verzerrung der Kurven, wie in

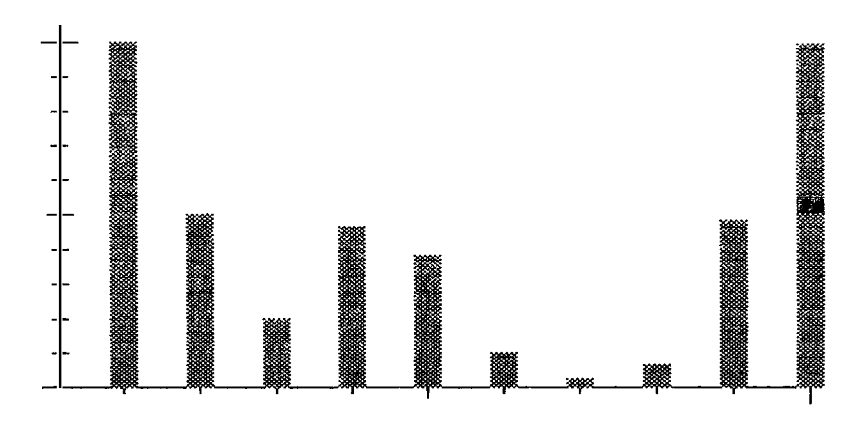
Abschnitt 3.1.1 dargelegt. Besonders hier spielen die Einheiten an den beiden Koordinatenachsen eine Rolle und sollten beachtet werden (s.u.).

b) *Balkendiagramme:*

Bei den Balkendiagrammen wird die Sache schon etwas komplizierter, obwohl das Prinzip identisch ist. Sie verwendet man bei relativ kleinen Datenmengen, die optisch besser sichtbar gemacht werden sollen. Ein Wertepaar dient nun nicht zur Bestimmung der Lage eines Punktes auf dem Bildschirm, sondern der Höhe und Position eines Balkens, der sich von der x-Achse in die Höhe zieht. Das folgende kleine Programm soll Ihnen die Programmierung solcher Balken demonstrieren. Nehmen wir an, es sollen die Umsatzzahlen einer (den Marktschwankungen recht stark ausgelieferten) Firma über 10 Jahre mit Hilfe eines Balkendiagramms dargestellt werden, um dem Leiter eine dringend nötige Übersicht zu vermitteln. Die jeweiligen Zahlen (in Tausend DM) werden dabei in DATA-Zeilen bereit gehalten. Sie könnten aber auch per INPUT solche Daten erfragen und evt. auf Diskette (Kasette) speichern und schon hätten Sie ein recht schönes Statistikprogramm.

```
100 REM ***********************
110 REM **                   **
120 REM **  BALKENDIAGRAMM   **
130 REM **                   **
140 REM ***********************
150 REM
320 REM
330 REM **** ACHSEN ****
340 REM      ******
350 GMODE 0,1 : GCLEAR : SCOL=0,7: SCOL=1,10 : REM GRAPHIK INIT
360 READ ZA : REM ZAHL DER WERTEPAARE
370 READ HI : REM HOECHSTER Y-WERT
380 XH = 300 : YH = 180 : REM ANZAHL DER PUNKTE IN X-/Y-RICHTUNG (MAX.)
390 XE=INT(XH/ZA*1) : YE=INT(YH/HI*10) : REM BERECHNUNG DER (1ER) 10ER E
INHEITEN
400 PLOT ,10,10 TO 10,190 : REM Y-ACHSE
410 PLOT ,10,190 TO 310,190 : REM X-ACHSE
420 T=-1:FOR Y=190 TO 10 STEP -YE
430 T=T+1:IF T/5-INT(T/5)=0 THEN:PLOT ,5,Y TO 15,Y : REM GROSSER S TRICH
435 IF T/10-INT(T/10)=0 THEN:PLOT ,3,Y TO 17,Y : REM GROSSER STRIC H
440 PLOT ,7,Y TO 13,Y : REM EINHEITSMARKIERUNGEN
450 NEXT Y
460 T=0:BE=20+XE/2 : REM BEGINN ERSTER STRICH
470 FOR X=BE TO 310 STEP XE
```

```
480 T=T+1:IF T/5-INT(T/5)=0 THEN:PLOT ,X,195 TO X,190 : REM GROSSE R
STRICH
485 IF T/10-INT(T/10)=0 THEN:PLOT ,X,198 TO X,190 : REM GROSSER ST RICH
490 PLOT ,X,193 TO X,190 : REM EINHEITSMARKIERUNGEN
500 NEXT X
600 REM
610 REM **** DIAGRAMM ****
620 REM      ********
630 BR=XE-20 : REM BALKENBREITE ERECHNEN
640 PO=BE-BR/2 : REM STARTPOSITION ERSTER BALKEN
650 FOR T=1 TO ZA
660 READ DA : REM DATEN LESEN
670 Y=190-DA*YE/10 : REM BALKENHOEHE (EINHEIT!)
690 FILL 255,PO,Y TO PO+BR,190 : REM BALKEN ZEICHNEN
710 PO=PO+XE : REM NEUE BALKENPOSITION
720 NEXT T
800 POKE198,0:WAIT 198,255:GMODE ,0:END
900 REM
910 REM **** DATEN ****
920 REM      *****
1000 DATA 10 : REM ANZAHL WERTE
1010 DATA 100 : REM HOECHSTER WERT
1100 DATA 100,50,20,46,38,10,2,6,48,99
```



Das Ganze sieht zunächst recht kompliziert aus, da eine Menge von Formeln angewandt werden. Diese "Formeln" sind aber recht gut zu verstehen, da Sie lediglich die Formatierung der Achsen und Balken betreffen.

Wir haben uns in diesem Programm als Ziel gestellt, möglichst variabel bezüglich der Anzahl und der Größe der Daten zu sein, ohne Bereichsüberschreitungen oder viel zu kleine Graphiken zu erzeugen. Aus diesem Grunde werden vor die eigentlichen Daten zwei Werte gestellt: die Anzahl der Daten und das größte Glied,

die in den Zeilen 360/370 eingelesen werden. Weiterhin legen wir uns den Bereich fest, in dem die Balkendiagramme liegen sollen. Wir haben dafür ein 300x180-Punktefeld reserviert (Z. 380).

Als erstes Problem stellt sich das Zeichnen der Achsen, da wir Einheiten eintragen wollen. Die Senkrechte (y-Achse) soll (so unsere Vereinbarung) jede 10. Einheit mit einem einfachen Strich anzeigen. Jede 50. Einheit steht ein etwa doppelt, jede 100. ein etwa dreimal so langer Strich. Auf der Waagerechten (x-Achse) wird die gleiche Einteilung unternommen, mit dem Unterschied, daß hier die 1., 5. und 10. Einheit hervorgehoben werden.

Dazu berechnen wir zunächst die Anzahl der Punkte, die auf eine (bei der x-Achse) bzw. 10 (y-Achse) Einheiten fallen dürfen, so daß wir sowohl nach rechts als auch nach oben nicht zu weit hinauskommen, aber auch die gesamte verwendbare Fläche ausnutzen (Z. 390).

Nun zeichnen wir erst einmal die bloßen Achsen (Z. 400/410). Dann folgt der Eintrag der Scalen (Z. 420-500). Dieser Vorgang sollte relativ leicht zu durchschauen sein. Dazu einige Bemerkungen: In den Zeilen 430, 435, 490 und 495 wird jeweils geprüft, ob gerade der 5. bzw. 10. Strich gezeichnet wird und entsprechend verlängert. Der dazu hinter dem IF stehende Ausdruck ist nichts weiter als das Abschneiden der Zahl vor dem Komma (auch Fraction genannt - das "Gegenteil" von INT). In Zeile 460 wird der Startpunkt der x-Scalierung errechnet. Die hier angeführte Formel hat etwas mit der Breite der Balken zu tun.

Jetzt erst werden die eigentlichen Balken gezeichnet. Dazu wird die Balkenbreite so bestimmt, daß zwischen zwei Balken genügend Platz ist (Z. 630). Die Formel für die Startpositon der Balken und Ihre Höhe sollten Sie ebenfalls verstehen (bei letzterer Formel wird mit der Anzahl der Punkte pro Einheit multipliziert, um der Scala an der y-Achse gerecht zu werden).

Sie können beliebig die Daten ab Zeile 1000 verändern, sollten aber darauf achten, keine negativen und nicht zu viele Daten zu verwenden. Bei extrem hohen Werten sollten Sie die Scaleneinheiten der Achsen ändern.

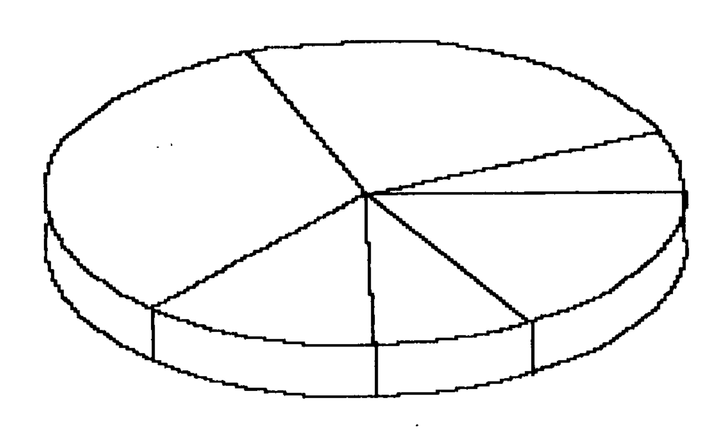
Damit haben Sie einen Einblick in die Programmiertechniken dieser Anwendung. Sicher fallen Ihnen noch viele Änderungen ein, die Sie an diesem Programm vornehmen können.

c) *Kuchendiagramme:*

Diese Art der graphischen Statistik ist geeignet zur übersichtlichen Darstellung von Ver- bzw. Aufteilungen von Mengen in Teilmengen und der Größenverhältnisse unter ihnen. Hierzu wird ein Kuchen (Kreis) gezeichnet, der die Gesamtheit aller zu betrachtenden Elemente darstellt (100 %) und der sich in verschieden große Teilstücke unterteilt. Die Größe der Teilstücke gibt dann den Anteil dieser Teilmenge an der Gesamtmenge an.

Oft sehen wir solche Graphiken bei Wahlen zur Darstellung z.B. der Sitzverteilung im Bundestag etc. oder im Geographieatlas, um beispielsweise die Anteile bestimmter Import- oder Exportwaren eines Landes am Gesamtumschlag zu demonstrieren usw.

Doch wie ist solches programmtechnisch zu verwirklichen? Schließlich haben wir es mit der ziemlich komplizierten Relation eines Kreises zu tun, der in bestimmte Winkelausschnitte unterteilt werden muß. In Kapitel 5.2.2.3 werden wir noch die Erzeugung eines Kreises (Ellipse) kennenlernen. Doch die dort angegebene Kreisformel ist nicht dazu geeignet lediglich Ausschnitte, die ja durch Angabe des jeweiligen Winkels definiert werden, zu zeichnen. Wir könnten hier beispielsweise den Supergraphik-Befehl CIRCLE verwenden. Doch wir wollen ja weiterlernen. Aus diesem Grunde weisen wir Sie hier in die Kreiserzeugung per Polarkoordinaten ein. Polarkoordinaten sind eine alternative Möglichkeit der Darstellung von Funktionen. Verwendet wird ein sogenanntes Polarkoordinatensystem, in dem nicht x und y als Achsenabschnitte, sondern w als Winkel zwischen der Verbindung von einem beliebigen, gesuchten Punkt zu dem Nullpunkt und der Waagerechten und l für den Abstand des Nullpunktes und dem besagten Punkt angegeben werden. Veranschaulicht sähe das dann etwa so aus:



Ein Kreis ist dann leicht durch die Änderung des Winkels w bei konstanthalten des Abstandes l zu konstruieren. Allgemein für eine beliebige Ellipse mit dem Mittelpunkt im Ursprung (0,0) gilt dann unter Berücksichtigung der Umrechnung von Polar- (l,w) in die uns bekannten sogenannten kartesischen Koordinaten (x,y):

```
x = a*cos(w)
y = b*sin(w)
```

wobei außer den genannten Parametern bedeuten:

a: Radius der Ellipse in x-Richtung
b: Radius der Ellipse in y-Richtung

Diese Zuordnung ist uns schon aus dem oben genannten Abschnitt bekannt. Mithilfe dieser beiden Formeln ist es uns nun möglich, einen bestimmten Randpunkt einer Ellipse durch Angabe des Winkels der Polarkoordinaten des Punktes zu bestimmen.

Eine Sache muß noch erläutert werden. Auch in den vorigen Abschnitten war öfter von Winkeln die Rede. Es gibt verschiedene Möglichkeiten, einen Winkel anzugeben:

- Angabe in Altgrad (0-360)
- Angabe in Neugrad (0-400)
- Angabe in Radiant (0-2*pi)

Die uns vertrauteste von allen ist wohl die erste. Doch unser Rechner rechnet nur mit der dritten, bei der 360 Altgrad dem Wert von 2*pi (pi=3.1415...) entsprechen. Dieser Wert entspricht der Länge eines Kreisbogens mit dem Radius 1 über den angegebenen Winkel. Wollen Sie Radiant in Altgrad oder umgekehrt umrechnen, so verwenden Sie diese Formel:

```
grad = 180*2*pi/rad        oder
rad  = 180*2*pi/grad
```

Damit besitzen wir das notwendige Rüstzeug, um das folgende Programm verstehen zu können. Es wurde mit dem Graphik-Paket aus dem 5. Kapitel geschrieben, es läuft also nicht mit der Supergraphik, sondern erst nach dem Starten dieses Graphik-Paketes!

```
100 REM ************************
110 REM **                    **
120 REM **  KUCHENDIAGRAMME   **
130 REM **                    **
140 REM ************************
150 REM
230 REM **** GRAPHIK-ROUTINEN ****
240 REM      *****************
250 IN=51200 : OF=51203 :REM INIT  /GRAPHIK OFF
260 GC=51206 : SC=51209 :REM GCLEAR/SET COLOR
270 PC=51212 : PL=51215 :REM PCOLOR/PLOT
280 UP=51218 : SL=51221 :REM UNPLOT/SET LINE
290 CL=51224 : GL=51227 :REM CLINE /GLOAD
300 GS=51230 : HC=51233 :REM GSAVE /HARDCOPY
320 REM
330 REM **** ELLIPSE ****
340 REM      *******
350 PI=3.1415
360 SYS IN : SYS GC : SYS SC,16*5+13 : REM GRAPHIK INIT
370 A = 100 : B = 60 : V1=160 : V2=80 : REM PARAMETER
380 W=0:GOSUB 930:X1=X:Y1=Y: REM X1 UND X2 VORBESTIMMEN
390 SP=7*PI/180 : REM STEP
400 BE=0:EN=2*PI : REM START- UND ENDWINKEL
410 GOSUB 800 : REM ELLIPSE ZEICHNEN
420 BE=0:EN=1.03*PI : REM ETWA 180 GRAD
430 V2=100 : REM TIEFER SETZEN
440 GOSUB 800 : REM ELLIPSE ZEICHNEN
500 REM
510 REM **** KUCHENSTUECKE ****
520 REM      **************
530 READ ZA : DIM T(ZA) : REM ANZAHL DER TEILE
540 FOR S=1 TO ZA
550 READ T(S) : REM DATEN LESEN
560 SU=SU+T(S) : REM SUMME BILDEN
570 NEXT S
580 W = 0 : REM STARTWINKEL
590 FOR S=1 TO ZA
600 PR=T(S)/SU : REM PROZENT AUSRECHNEN
610 WA=2*PI*PR : REM WINKEL DES AUSSCHNITTES
620 W=W+WA : REM WIRKLICHER WINKEL
630 V2=80:GOSUB 930:REM KOORDINATEN ERRECHNEN
640 SYS SL,V1,V2,X,Y : REM TRENNLINIE
650 IF W>PI THEN 690 : REM NUR AUF DER SICHTBAREN SEITE
660 X1=X:Y1=Y : REM MERKEN
670 V2=100:GOSUB 930:REM UNTERE KOORD. ERRECHNEN
680 SYS SL,X1,Y1,X,Y : REM SENKRECHTE ZUM UNTEREN BOGEN
690 NEXT S
799 WAIT 198,255:SYS OF:END
800 REM
810 REM **** ELLIPSENBOGEN ****
820 REM      **************
```

```
830 FOR W=BE TO EN+SP STEP SP : REM WINKEL BESTIMMEN
840 GOSUB 930 : REM KOORDINATEN BESTIMMEN
850 SYS SL,X1,Y1,X,Y : REM LINIE
860 X1=X:Y1=Y
870 NEXT W : RETURN
900 REM
910 REM **** PUNKT BERECHNEN ****
920 REM      ****************
930 X = A*COS(W) + V1
940 Y = B*SIN(W) + V2 : RETURN
1000 REM
1010 REM **** DATEN ****
1020 REM      *****
1100 DATA 6 : REM ANZAHL
1110 DATA 20,10,15,40,30,8
```

Wie Sie sehen, können Sie hier aus den Zeilen 800-940 für Ihre eigenen Programme eine alternative Kreisformel entnehmen. Die Übergabeparameter sind im einzelnen:

BE: Startwinkel
EN: Endwinkel
SP: Schritteinheit
V1: Mittelpunkt-x-Koordinate
V2: Mittelpunkt-y-Koordinate
A: x-Radius
B: y-Radius

Zu der Schrittweite SP muß folgendes gesagt werden: Sie bestimmt, in welchem Winkelabstand die einzelnen Punkte der Ellipse berechnet werden, also die Genauigkeit. Diese Punkte werden dann in Zeile 850 durch eine Linie verbunden. Wählen Sie SP sehr groß, z.B. 30 oder 45, so erhalten Sie spezielle Effekte: Damit wird dann ein gleichseitiges Vieleck (8-Eck etc.) gezeichnet, was für andere Zeichnungen gut verwendet werden kann!

Eigentlich muß die Zeile 380 bereits in diesem Unterprogramm aufgeführt werden und sollte auch von Ihnen übernommen werden, sofern Sie lediglich die Ellipsenroutine verwenden wollen. Hier wurde sie herausgenommen, um einen speziellen Effekt zu erzielen (s.u.).

In dem obigen Programm wollen wir nicht einfach eine simple Scheibe zeichnen und dann entsprechende Abschnitte abtragen. Vielmehr soll das Ganze etwas Format haben und in 3-D gezeichnet sein. Es wird eine runde Platte mit einer gewissen Dicke dargestellt, die in die unterschiedlichen "Kuchenstücke" zerschnitten ist und von schräg oben gesehen werden soll. Dazu zeichnen wir zunächst einmal eine Ellipse als Oberfläche (Z. 410). Alsdann zeichnen wir die gleiche Figur nur um ein paar Punkte nach unten verschoben und als Halbellipse, um den unteren Rand der Platte darzustellen (Z. 440). Eigentlich wird bei dieser Konstruktion der Platte ein wenig "gemogelt", da wir die seitlichen Ränder als gerade Linien darstellen müßten (was rechts auch geschieht), dafür zeichnen wir die Ellipse ein klein wenig über 180 Grad (1*pi) hinaus, was wegen der Auflösung kaum auffällt.

Nun kommen wir zu den eigentlichen Kuchenschnitten:
Vorher sollten wir etwas über die verwendete Datenstruktur sagen. Am Anfang vor den eigentlichen Daten steht wie immer ihre Anzahl. Dann folgen beliebig viele und beliebig hohe Zahlen, die z.B. die Anzahl der Sitze der einzelnen Parteien oder allgemein die Größe der verschiedenen Teilmengen wiedergeben.

Im ersten Teil der ab Zeile 500 folgenden Routine werden alle durch ZA angegebenen Daten in ein eindimensionales Feld eingelesen und (das ist der eigentliche Grund für diese Schleife) die Summe aller Teilmengen gebildet (Z. 540-570).

In der nächsten FOR...NEXT-Schleife werden dann die Unterteilungen vorgenommen: Zeile 600 rechnet den prozentualen Anteil der jeweiligen Partei an der Gesamtheit aus. Zeile 610 bestimmt die daraus resultierende Größe des Kreis- (Ellipsen-) Ausschnittes in Radiant, die zu dem aktuellen Winkel W hinzugezählt wird (Z. 620). Nun, da der Winkel des betreffenden Schnittes bekannt ist, kann die Position des entsprechenden Randpunktes der Ellipse errechnet werden. Alsdann wird vom Mittelpunkt der Ellipse zu diesem Punkt eine Linie gezeichnet (Z.640).

Nun kommt es darauf an, ob sich dieser Schnitt bereits auf der hinteren Hälfte der Scheibe oder noch vorne befindet. Im ersten Fall ist die Sache erledigt und der nächste Wert kann bearbeitet werden. Im zweiten Fall jedoch muß der Schnitt noch sichtbar bis zur Unterkante gezogen werden. Dies geschieht, indem wir einfach die Position auf der verschobenen Kante berechnen und vom zuletzt gezeichneten Punkt bis hierhin eine Linie ziehen (Z. 650-680). Damit wäre der Dreidimensionalität Genüge getan.

Auch dieses Programm ist selbstverständlich voll ausbaufähig. Sie könnten beispielsweise die einzelnen Teile in verschiedenen Farben oder schraffiert zeichnen usw. Ihrer Phantasie sind keine Grenzen gesetzt.

3.2 Laufschriften

Eine verlockend einfache und gleichzeitig recht schöne Art und Weise, beliebig gestaltete und bewegliche Buchstaben auf den Bildschirm (auch in die hochauflösende Graphik) zu bringen, ist die Konstruktion von Buchstabensprites. Die Sprites besitzen eine geradezu phantastische Auflösung zur Erstellung von Schrift und können vergrößert, bewegt usw. werden.

Wir werden hier zwei Versionen eines Programmes vorstellen. Die erste Version arbeitet mit den Sprite-Befehlen der Supergraphik, die zweite arbeitet mit dem normalen BASIC ohne jede Erweiterung, um Ihnen bereits eine kleine Vorstellung von den Problemen der Sprite-Programmierung ohne BASIC-Erweiterung zu geben. Sie brauchen hier an dieser Stelle noch nicht unbedingt alles zu verstehen, da hierzu die Lektüre der folgenden Kapitel notwendig ist. Überschlagen Sie also ruhig die in kursiv gedruckten Erläuterungen, bis Sie das hierzu notwendige Wissen besitzen.

Das Problem bei der Bewältigung der Aufgabe in originalem BASIC ist der begrenzte Speicherraum, der uns unter BASIC zur Verfügung steht. Mit den vier Blöcken, die uns in BASIC zur völlig freien Verfügung stehen, kommen wir nicht aus. Um hier Abhilfe zu schaffen, können wir erstens den gesamten VIC-Adreßbereich um 16, 32 oder gar 48 K nach oben verschieben (s. Kap. 4.3.2) und so in Speicherebenen gelangen, die wir ohne weiteres nutzen können. Doch hier tritt eine kleine Schwierigkeit auf. Es verschieben sich ja nicht nur die Spriteblöcke, sondern gleich alle Bildschirmspeicher wie Video-RAM und Graphikspeicher, was die Handhabung erheblich erschwert.

Wir wollen uns die zweite Möglichkeit zu Nutze machen: Wir packen alle Sprite-Definitionen, die wir in unserem Programm verwenden, in den Bereich von $2000-$3FFF (8192-16383), wo wir normalerweise unsere Graphik beherbergen. Zwar können wir dann nicht mehr gleichzeitig Graphik anzeigen, aber man muß bereit sein auch Kom-

promisse zu schließen. Damit verwenden wir die Blöcke 8192/64=128 bis 255 und haben Platz für insgesamt 127 Spritedefinitionen, was ausreichend sein sollte. Wir brauchen dann nur darauf zu achten, daß wir nicht unbedingt riesige Mammutprogramme oder solche mit einem großen Speicheraufwand schreiben, und der Fall ist erledigt.

Kommen wir zu den Buchstabendefinitionen. Normalerweise ist es unnötiger Platzverbrauch, alle Spritedefinitionen als DATA-Zeilen in unser BASIC-Programm niederzulegen. Alternative Methoden werden Ihnen in Kapitel 5.3 vorgestellt. Hier allerdings wollen wir ausnahmsweise damit arbeiten, zumal wir nur wenige Buchstabendefinitionen vorstellen.

Die einzelnen Sprites können Sie mit dem Spriteformer in Abschnitt 5.3 erstellen und später z.B. direkt in den Speicher einlesen. Sie können sich so einen ganzen Vorrat an Zeichen erstellen und bei Bedarf abrufbereit halten.

Wir wollen aber nun zu der eigentlichen Aufgabe vorschreiten: der Programmierung von Laufschriften. Hierzu ein kleines BASIC-Programm (zunächst mit der Supergraphik):

```
100 REM ***********************
110 REM **                   **
120 REM **  LAUFSCHRIFTEN    **
130 REM **                   **
140 REM ***********************
150 REM
160 GMODE 0,1:GCLEAR
170 COLOR= 0,0 :REM RAHMEN UND HINTERGRUND = SCHWARZ
180 DIM A$(96) :REM SPRITEDEFINITIONEN
190 REM
200 REM
210 REM **** EINSPEICHERUNG ****
220 REM      ***************
230 REM
240 FOR B=1 TO 9  :REM 9 BUCHSTABEN
250 READ A$       :REM NAME DES BUCHSTABEN
260 BK=ASC(A$)-32  :REM SPRITES NACH ASCII GEORDNET IM SPEICHER (BL
OCKNUMMER)
270 SREAD A$(BK)  :REM SPRITE-DATEN LESEN
280 NEXT B        :REM 9 BUCHSTABEN LESEN
290 REM
300 REM
310 REM **** INITIALISIERUNG ****
320 REM      ***************
330 REM
```

```
340 FOR S=0 TO 7
350 SDEFINE S,""  :REM SPRITES AUS
360 SMODE S,3,S+1 :REM FARBE UND GROESSE FESTLEGEN
370 NEXT S
380 REM
390 REM
400 SP=0          :REM START-SPRITENUMMER
410 ZA=8 :REM ANZAHL DER BUCHSTABEN AUF DEM BILD (GLEICHZEITIG / MAX.:8)
420 AB=330/ZA     :REM ABSTAND ZWEIER SPRITES
430 GE = 20       :REM GESCHWINDIGKEIT
440 REM
450 REM
460 REM **** LAUFSCHRIFT ****
470 REM      ************
480 REM
490 TE$="DATA-BECKER---" :REM LAUFTEXT
500 FOR LA=1 TO 10       :REM 10 DURCHLAEUFE
510 FOR BU=1 TO LEN(TE$)
520 BU$=MID$(TE$,BU,1)   :REM LAUFENDER BUCHSTABE
530 BK =ASC(BU$)-32      :REM NUMMER DES BUCHSTABEN
540 XK(SP)=351
550 YK(SP)=100           :REM SPRITE-KOORDINATEN
560 SSET SP,XK(SP),YK(SP):REM SPRITE-KOORDINATEN
570 SDEFINE SP,A$(BK)    :REM SPRITE EINSCHALTEN
580 IF SO<>ZA-1 THENSO=SP
590 SP=SP+1              :REM NAECHSTE SPRITENUMMER
600 IF SP=ZA THENSP=0
610 REM
620 REM BEWEGEN:
630 REM
640 FOR K=1 TO AB STEP GE:REM AB SCHRITTE BEWEGEN
650 FOR S=0 TO SO        :REM SO SPRITES BEWEGEN
660 XK(S)=XK(S)-GE       :REM NEUE X-KOORDINATEN
670 IF XK(S)<0 THEN:SDEFINE S,"" : GOTO690:REM SPRITE AUS
680 SSET S,XK(S),YK(S)   :REM SPRITE POSITIONIEREN
690 NEXT S,K
700 NEXT BU :REM NAECHSTEN BUCHSTABEN
710 NEXT LA :REM NAECHSTEN DURCHLAUF
720 REM
730 REM
740 REM **** SPRITE-DATA ****
750 REM      ************
760 REM
770 DATA A : REM BUCHSTABE A
780 DATA 000,000,000, 000,000,000, 003,255,192
790 DATA 007,000,224, 006,000,096, 006,000,096
800 DATA 006,000,096, 006,000,096, 006,000,096
810 DATA 006,000,096, 007,255,224, 006,000,096
820 DATA 006,000,096, 006,000,096, 006,000,096
830 DATA 006,000,096, 006,000,096, 006,000,096
840 DATA 006,000,096, 000,000,000, 000,000,000
```

```
850 REM
860 DATA B : REM BUCHSTABE B
870 DATA 000,000,000, 000,000,000, 003,255,000
880 DATA 003,001,128, 003,000,192, 003,000,192
890 DATA 003,000,192, 003,000,192, 003,001,128
900 DATA 003,255,000, 003,001,128, 003,000,192
910 DATA 003,000,096, 003,000,096, 003,000,096
920 DATA 003,000,096, 003,000,192, 003,001,128
930 DATA 003,255,000, 000,000,000, 000,000,000
940 REM
950 DATA C : REM BUCHSTABE C
960 DATA 000,000,000, 000,000,000, 001,255,192
970 DATA 003,129,192, 003,000,000, 003,000,000
980 DATA 003,000,000, 003,000,000, 003,000,000
990 DATA 003,000,000, 003,000,000, 003,000,000
1000 DATA 003,000,000, 003,000,000, 003,000,000
1010 DATA 003,000,000, 003,000,000, 003,129,192
1020 DATA 001,255,192, 000,000,000, 000,000,000
1030 REM
1040 DATA D : REM BUCHSTABE D
1050 DATA 000,000,000, 000,000,000, 015,255,000
1060 DATA 012,001,128, 012,000,192, 012,000,192
1070 DATA 012,000,192, 012,000,192, 012,000,192
1080 DATA 012,000,192, 012,000,192, 012,000,192
1090 DATA 012,000,192, 012,000,192, 012,000,192
1100 DATA 012,000,192, 012,000,192, 012,001,128
1110 DATA 015,255,000, 000,000,000, 000,000,000
1120 DATA E : REM BUCHSTABE E
1130 DATA 000,000,000, 000,000,000, 003,255,192
1140 DATA 003,001,192, 003,000,000, 003,000,000
1150 DATA 003,000,000, 003,000,000, 003,006,000
1160 DATA 003,255,000, 003,006,000, 003,000,000
1170 DATA 003,000,000, 003,000,000, 003,000,000
1180 DATA 003,000,000, 003,000,000, 003,001,192
1190 DATA 003,255,192, 000,000,000, 000,000,000
1200 REM
1210 DATA K : REM BUCHSTABE K
1220 DATA 000,000,000, 000,000,000, 003,001,128
1230 DATA 003,003,000, 003,006,000, 003,012,000
1240 DATA 003,024,000, 003,048,000, 003,096,000
1250 DATA 003,192,000, 003,192,000, 003,096,000
1260 DATA 003,048,000, 003,024,000, 003,012,000
1270 DATA 003,006,000, 003,003,000, 003,001,000
1280 DATA 003,000,192, 000,000,000, 000,000,000
1290 REM
1300 DATA R : REM BUCHSTABE R
1310 DATA 000,000,000, 000,000,000, 001,255,000
1320 DATA 003,001,128, 003,000,192, 003,000,192
1330 DATA 003,000,192, 003,000,192, 003,001,128
1340 DATA 003,255,000, 003,192,000, 003,096,000
1350 DATA 003,048,000, 003,024,000, 003,012,000
```

```
1360 DATA 003,006,000, 003,003,000, 003,001,000
1370 DATA 003,000,192, 000,000,000, 000,000,000
1380 REM
1390 DATA T : REM BUCHSTABE T
1400 DATA 000,000,000, 000,000,000, 015,255,192
1410 DATA 012,048,192, 000,048,000, 000,048,000
1420 DATA 000,048,000, 000,048,000, 000,048,000
1430 DATA 000,048,000, 000,048,000, 000,048,000
1440 DATA 000,048,000, 000,048,000, 000,048,000
1450 DATA 000,048,000, 000,048,000, 000,048,000
1460 DATA 000,048,000, 000,000,000, 000,000,000
1470 REM
1480 DATA "-" : REM BINDESTRICH
1490 DATA 000,000,000, 000,000,000, 000,000,000
1500 DATA 000,000,000, 000,000,000, 000,000,000
1510 DATA 000,000,000, 000,000,000, 000,000,000
1520 DATA 000,000,000, 003,255,192, 000,000,000
1530 DATA 000,000,000, 000,000,000, 000,000,000
1540 DATA 000,000,000, 000,000,000, 000,000,000
1550 DATA 000,000,000, 000,000,000, 000,000,000
```

Nun das gleiche Programm im originalen BASIC (die DATA-Zeilen sind hierbei identisch und wurden aus Platzersparnis weggelassen):

```
100 REM **********************
110 REM **                  **
120 REM **  LAUFSCHRIFTEN   **
130 REM **                  **
140 REM **********************
150 REM
160 V = 53248 : REM BASISADRESSE VIDEOCONTROLLER
170 POKE V+32,0 : POKE V+33,0 : REM RAHMEN UND HINTERGRUND = SCHWARZ
175 PRINT CHR$(147)
180 REM
190 REM **** EINSPEICHERUNG ****
200 REM      ***************
210 FOR X=1 TO 9 : REM 9 BUCHSTABEN
220 READ A$ : REM NAME DES BUCHSTABEN
230 BK=ASC(A$)-32+128 : REM SPRITES NACH ASCII GEORDNET IM SPEICHER
(BLOCKNUMMER)
240 AD=BK*64 : REM ADRESSEN AB 8192
250 FOR Y=AD TO AD+62
260 READ DA : POKE Y,DA : REM DATEN LESEN UND SCHREIBEN
270 NEXT Y,X : REM 63 DATEN/8 BUCHSTABEN LESEN
300 REM
```

```
310 REM **** INITIALISIERUNG ****
320 REM      ****************
330 FOR X=0 TO 7
340 POKE V+39+X,X+1 : REM FARBEN SPRITES 0-7 FESTLEGEN
350 NEXT X
360 POKE V+23,255 : REM ALLE SPRITES GROSS (Y)
370 POKE V+29,255 : REM ALLE SPRITES GROSS (X)
380 POKE V+27,0 : REM PRIORITAET
390 POKE V+28,0 : REM NORMAL-FARBEN
400 SP=0 : REM START-SPRITENUMMER
410 ZA=8 : REM ANZAHL DER BUCHSTABEN AUF DEM BILD (GLEICHZEITIG /
MAX.:8)
420 AB=330/ZA : REM ABSTAND ZWEIER SPRITES
430 GE = 20 : REM GESCHWINDIGKEIT
500 REM
510 REM **** LAUFSCHRIFT ****
520 REM      ************
530 TE$="DATA-BECKER---" : REM LAUFTEXT
540 FOR LA=1 TO 10 : REM 10 DURCHLAEUFE
550 FOR BU=1 TO LEN(TE$)
560 BU$=MID$(TE$,BU,1) : REM LAUFENDER BUCHSTABE
570 BK =ASC(BU$)-32 + 128 : REM BLOCKNUMMER DES BUCHSTABEN
580 POKE 2040+SP,BK : REM SPRITE AUF ENTSPRECHENDEN BLOCK SETZEN
590 POKE V+SP*2,95 : REM SPRITE-X-KOORD.-LOW-BYTE
600 POKE V+SP*2+1,100 : REM SPRITE-Y-KOORD.
610 POKE V+16,PEEK(V+16) OR 2^SP : REM SPRITE-X-KOORD.-HIGH-BIT
620 POKE V+21,PEEK(V+21) OR 2^SP : REM SPRITE EINSCHALTEN
630 SP=SP+1 : REM NAECHSTE SPRITENUMMER
635 IF SP=ZA THEN SP=0
640 FOR K=1 TO AB STEP GE : REM AB SCHRITTE ***  BEWEGEN  ***
650 FOR S=0 TO ZA-1 : REM ZA SPRITES BEWEGEN
660  AD=V+S*2:XL=PEEK(AD)-GE:XH=PEEK(V+16)AND2^S:REM  X-KOORDINATE  DES
SPRITES
670  IF  XL<0  THEN  XL=256+XL:POKE  V+16,PEEK(V+16)  AND  255-
2^S:IFXH=0THENXL=0
680 POKE AD,XL
685 GOSUB 800
690 NEXT S,K
700 NEXT BU : REM NAECHSTEN BUCHSTABEN
710 NEXT LA : REM NAECHSTEN DURCHLAUF
800 POKE V+21,PEEK(V+21) AND 255-2^SP:RETURN : REM SPRITE AUS
1000 REM
1010 REM **** SPRITE-DATA ****
1020 REM      ************
1090 REM
```

Dieses Programm kann Ihnen nur die Grundzüge der Laufschrifttechnik vermitteln. Es liegt an Ihnen, die entsprechenden Anwenderprogramme zu schreiben. Richtig schnell und ansehnlich wird das Ganze natürlich erst in Maschinensprache. Aber ich hoffe, Sie haben auch so Ihren Spaß daran. Wenn Sie sich das Programm durchschauen, so sollte Ihnen das Verständnis der Abläufe durch die vielen REM-Zeilen gut verständlich sein. Wie gesagt können Sie sich einen ganzen Zeichensatz zusammenstellen und sclíließ lich beliebige Texte ausgeben. Viel Spaß!

3.3 Das Geheimnis der Spiele

Inzwischen gibt es wohl bereits annähernd tausend gute bis weniger gute Spiele für den Commodore 64, die in Computer-Shops zu kaufen sind. Wir lassen uns von ihrer Graphik, ihren Soundkaskaden berauschen und klopfen uns befriedigt auf die Schulter: "Hab' ich mir doch einen guten Rechner gekauft, was der alles kann!". In der Tat zeigen von allen Programmen ganz besonders die Spiele, welche Qualitäten ein Gerät besitzt, da diese meist bis an die Grenzen des Machbaren stoßen. Beim 64er sind diese Grenzen ziemlich hoch angesetzt und sogar die besten Spieleprogrammierer haben Probleme alle Möglichkeiten voll auszunutzen. Doch was nutzt einem ein guter Computer, mit dem alles möglich ist, wenn man selbst nicht weiß, wie es geht? Und ewig nur zuzuschauen, was andere programmiert haben, macht einen auch nicht satt.

In diesem Buch haben Sie bereits vieles erfahren, das Ihnen helfen wird, Ihren Computer bezüglich Graphik, Sprites und allgemein der Bildschirmausgabe optimal zu nutzen. Natürlich konnte nicht annähernd alles, was der 64er bietet und im Kapitel 4 (Hardwaregrundlagen) steht, auch zur Anwendung gebracht werden. Das ist auch gut so. Auf diese Weise bleibt noch genügend Freiraum für Ihren Forscherdrang.

Die meisten Spiele sind in Maschinensprache geschrieben, da das originale BASIC einfach zu langsam ist. Es wurden Ihnen bereits einige Utilities in Assembler angeboten, die quasi als kleine Erweiterung des BASIC-Befehlssatzes für die Aufbesserung der Geschwindigkeiten Ihrer Programme zur Verfügung stehen (z.B. das Graphik-Paket). Hier nun sollen Ihnen einige Techniken und Erweiterungen speziell für Spiele vermittelt werden (sie sind selbstverständlich auch für andere Anwendungen recht nützlich). So sind Sie in der Lage, auch in BASIC (erst recht in Maschinensprache) schnelle und anspruchsvolle Spiele zu programmieren.

3.3.1 Animation

Unter Animation versteht man die Erzeugung bewegter Bilder auf dem Bildschirm. Natürlich "leben" die Spiele von der Animation, sofern nicht etwa Denkspiele wie Schach, Memorie etc. gemeint sind. Ohne Bewegung auf dem Bildschirm "läuft" nichts. Oft werden Spiele nur aufgrund der Qualität dieser Bewegung in die Reihe der Actionspiele oder "sonstige" Spiele eingereiht. Wir wollen uns deshalb zunächst mit diesem wichtigen Thema beschäftigen.

Man unterscheidet beim Commodore 64 fünf Arten von Animation:

- Spriteinterne Bewegung
- Spriteverschiebung
- Zeicheninterne Bewegung
- Zeichenverschiebung
- Graphische Animation

Unter 'intern' verstehen wir hier das Bewegen des jeweiligen Objektes selbst, ohne daß es seine Position auf dem Bildschirm verändert.

Die ersten beiden Formen haben wir bereits in den zwei Beispielen zur Spriteprogrammierung im Abschnitt 5.3.2 ausführlich dargelegt und erläutert. Sie sollten sich diesen Paragraphen sowieso durchgelesen haben, da die Spriteprogrammierung ein, wenn nicht gar das wichtigste, Fundament der Spiele darstellt.

Auch den letzten Punkt wollen wir hier nicht abhandeln, da er sich aus den verschiedenen Graphikkapiteln ableitet und aus Geschwindigkeitsgründen nur selten oder gar nicht bei Spielen Verwendung findet.

Äußerst beliebt sind dagegen die beiden restlichen Punkte. Meist werden sie in Verbindung mit einer Zeichensatzänderung verwendet.

a) Zeicheninterne Bewegung:

Das Prinzip der zeicheninternen Bewegung ist das gleiche, wie bei der Bewegung der Sprites. Eine aus einem oder mehreren Zeichen zusammengesetzte Figur wird durch den stetigen Wechsel von Zuständen bestimmter Teile des Objektes so verändert, daß daraus eine Bewegung entsteht. Im Klartext bedeutet das folgendes:

Angenommen wir wollen ein Männchen so steuern, daß es stets beide Arme und Beine auf und nieder bewegt. Wir setzen dieses Männchen dann aus mehreren Teilen zusammen, damit es nicht zu klein wird. Um nun die gewünschte Bewegung zu programmieren, legen wir uns jedoch zwei oder mehr Männchen bereit, die jeweils andere Phasen der Bewegung festhalten. Diese verschiedenen Figuren lassen wir dann in einem uns genehmen Takt abwechselnd an der gleichen Stelle auf dem Bildschirm erscheinen, und schon haben wir den gewünschten Effekt.

Besonders effektiv wird das Ganze natürlich mit einem veränderten Zeichensatz oder ein paar veränderten Zeichen. Wie dies auf einfache Weise gemacht wird, zeigt Ihnen Abschnitt 5.4. Besonders hier zeigen Multicolorzeichen ihren Sinn. Ihr Objekt wird recht schön farbig. Sie sehen, beim Thema Spiele fließt alles Wissen und Können des Programmierers zusammen.

Das folgende Beispiel soll Ihnen den Nutzen auch des originalen Zeichensatzes für dieses Thema darlegen:

```
100 REM ********************
110 REM **                **
120 REM **  ANIMATION-1   **
130 REM **                **
140 REM ********************
150 REM
160 A$(0)=" "     +CHR$(119)
170 A$(1)=CHR$( 99)+CHR$(123)+CHR$( 99)
180 A$(2)=CHR$(167)+CHR$(183)+CHR$(165)
190 A$(3)=" "     +CHR$(113)
200 A$(4)=CHR$(173)+CHR$(123)+CHR$(189)
210 A$(5)=CHR$(183)+CHR$(183)+CHR$(183)
300 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
310 X=18 : REM SPALTENPOSITION
320 FOR ZA=0 TO 5 STEP 3
330 Y=12 : GOSUB 1000 : REM POSITIONIEREN
340 PRINT A$(ZA):GOSUB 1010 : REM KOPF
350 PRINT A$(ZA+1):GOSUB 1010 : REM RUMPF/ARME
```

```
360 PRINT A$(ZA+2) : REM BEINE
370 FOR S=1 TO 100 : NEXT S : REM WARTESCHLEIFE
380 NEXT ZA
390 GOTO 320 : REM MIT <RUN/STOP> UNTERBRECHEN
400 REM
410 REM **** POSITIONIEREN ****
420 REM      **************
1000 PRINT CHR$(19);:IF Y>0 THEN FOR T=1 TO Y:PRINT:NEXT T
1010 PRINT TAB(X);: RETURNzln1
```

Die beiden letzten Zeilen lassen sich sehr einfach durch den Supergraphik-Befehl POS= ersetzen. Damit Sie aber nicht nur für diesen einen Befehl die Supergraphik laden müssen, haben wir ihn hier ersetzt.

In diesem Programm werden zu Anfang (Zeilen 160-210) die Teile für die zwei Phasen der Bewegung des Männchens definiert. Insgesamt bauen wir es aus 7 Zeichen auf, die in 3 untereinander liegenden Reihen stehen sollen. Jede der drei Reihen wird in einen separaten Array-Speicher (A$(...)) abgelegt (Nr. 0-2 für die erste Phase und Nr. 3-5 für die zweite). Alsdann werden die Startkoordinaten des Männchens festgelegt und positioniert (Z. 330). Der Rest ist relativ einfach zu durchschauen: Die drei Reihen werden gezeichnet (1.Phase), wobei jeweils nur ein Teil der Positionierungsroutine aufgerufen wird. Im zweiten Durchlauf der FOR...NEXT-Schleife werden dann die Reihen der 2. Phase gezeichnet. Durch Veränderung der Länge der Warteschleife in Zeile 370 kann die Geschwindigkeit der Bewegung gesteuert werden. Versuchen Sie doch einmal, dieses Männchen durch eigene Zeichendefinitionen darzustellen (s. Kap. 5.4). Dann ist es Ihnen gleichfalls möglich, mehr als zwei Bewegungsphasen nacheinander ablaufen zu lassen, was die Effektivität natürlich um einiges steigert.

b) Zeichenverschiebung:

Wie bei den Sprites können wir auch unser Männchen über den Bildschirm bewegen. Dies geschieht auf ähnliche Weise, wie in Kapitel 5.3 angegeben. Wir zeichnen unser Männchen an einer bestimmten Position auf den Bildschirm. Nach einiger Zeit (Warteschleife) löschen wir es wieder und setzen es dafür ein klein wenig verschoben weiter nach rechts, links, oben oder unten usw. Aus dieser stetigen Verschiebung resultiert dann eine kontinuierliche Bewegung, wie wir sie von den Sprites her kennen. Das einzige Problem dabei ist die Auflösung der Bewegung. Normalerweise können wir unser Objekt immer nur um minimal ein Zeichen verschieben. Dieses Manko können wir allerdings

ein klein wenig dadurch ausgleichen, daß wir auch hier "Zwischenphasen" programmieren (das Objekt steht praktisch "zwischen" zwei Zeichen). Das geht natürlich nur unter Veränderung des Zeichensatzes.

Doch beschäftigen wir uns lieber erst einmal mit den Grundlagen. Das folgende Programm vereinigt die Technik der internen Bewegung mit der Zeichenverschiebung und vermittelt ein schon recht hübsches Bild. Wenn wir dieses Männchen nun noch per Joystick steuern, bleibt (fast) kein Wunsch mehr offen.

```
100 REM ********************
110 REM **                **
120 REM **  ANIMATION-2   **
130 REM **                **
140 REM ********************
150 REM
160 A$(0)=" "+" "      +CHR$(119)+" "      +" "
170 A$(1)=" "+CHR$( 99)+CHR$(123)+CHR$( 99)+" "
180 A$(2)=" "+CHR$(167)+CHR$(183)+CHR$(165)+" "
190 A$(3)=" "+" "      +CHR$(113)+" "      +" "
200 A$(4)=" "+CHR$(173)+CHR$(123)+CHR$(189)+" "
210 A$(5)=" "+CHR$(183)+CHR$(183)+CHR$(183)+" "
300 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
305 SP=1:B=0:E=33
310 FOR X=B TO E STEP SP : REM SPALTENPOSITION
320 FOR ZA=0 TO 5 STEP 3
330 Y=12 : GOSUB 1000 : REM POSITIONIEREN
340 PRINT A$(ZA):GOSUB 1010 : REM KOPF
350 PRINT A$(ZA+1):GOSUB 1010 : REM RUMPF/ARME
360 PRINT A$(ZA+2) : REM BEINE
370 FOR S=1 TO 60 : NEXT S : REM WARTESCHLEIFE
380 X=X+SP
390 NEXT ZA
400 X=X-SP
410 NEXT X
420 SP=-SP : ZW=B : B=E : E=ZW : GOTO 310 : REM AUSTAUSCH (BEWEGUNGSR.
AENDERN)
500 REM
510 REM **** POSITIONIEREN ****
520 REM      **************
1000 PRINT CHR$(19);:IF Y>0 THEN FOR T=1 TO Y:PRINT:NEXT T
1010 PRINT TAB(X);: RETURN
```

Wir haben in diesem Programm um die bereits bekannte Bewegungsroutine eine weitere FOR...NEXT-Schleife gepackt. Weiterhin mußten wir die Reihendefinitionen in den Zeilen 160-210 jeweils um ein Leerzeichen vorne und hinten erweitern, um ein Löschen der zuletzt gezeichneten Figur sowohl beim Hin-, als auch beim Zurückgehen zu gewährleisten. Vielleicht versuchen Sie einmal hinter die Bedeutung der Speicher SP, B, E (und ZW) zu kommen. Sie dienen zur Bereitstellung der notwendigen Parameter für Rechts- und Linkslauf.

Damit haben wir Ihnen einige Tips gegeben, wie Sie neben den Sprites noch einige einfache Bewegungen mehr in Ihr Spiel bekommen. Kommen wir daher gleich zum nächsten Thema.

3.3.2 Scrolling

Wer hat nicht schon einmal 'Defender' oder ähnliche Actionspiele gesehen oder gar mit ihnen gespielt, in denen Sie Führer eines Raumschiffes sind, das mit einer Höllengeschwindigkeit durch den Weltraum fliegt. Oder Sie fahren mit einem Rennwagen unter Aufbringung aller Konzentration auf einer Piste, verfolgt von anderen Rivalen, die Sie abdrängen.

Doch wenn Sie genauer hinschauen, sind es nicht das Flugzeug oder das Auto, die sich vorwärts bewegen, sondern vielmehr der Hintergrund. D.h. der gesamte Bildschirm (oder Teile) wird nach rechts, links, oben oder unten verschoben, während das jeweilige zu steuernde Objekt (ein Sprite) meist nur beschränkt bewegt werden kann. Nun ist ein solches Verschieben, Rollen oder Scrolling des Bildschirms eine sehr zeitaufwendige Sache und kann daher nur in Maschinensprache durchgeführt werden. Dabei verwendet man nicht etwa Graphik (hier müßten für einmal verschieben 8 K bewegt werden) sondern den Textmodus, was bei Spielen fast das Gleiche ist, da wir ja den Zeichensatz beliebig verändern können und so quasi hochauflösende Bilder erhalten. Bekanntlich müssen hier nur etwa 1 K Bytes bewegt werden, was die Arbeit gewaltig verringert. Im folgenden werden Ihnen entsprechende Assemblerroutinen angegeben, die Sie genauso wie die Befehle des Graphik-Paketes per SYS aufrufen können:

```
10: CC00                  *= $CC00
20:                   ;
30:                   ;******************
40:                   ;**              **
50:                   ;**  SCROLLING  **
60:                   ;**              **
70:                   ;******************
80:                   ;
110: CC00             CHKGET =  $B7F1
120: CC00             OB     =  2038
130: CC00             UN     =  2039
140: CC00             FLAG   =  $FD
150: CC00             ZAHL   =  $FE
160: CC00             ADRESS =  $61
200: CC00 20 F1 B7 START  JSR CHKGET  ;KOMMA + BYTE HOLEN
210: CC03 8A              TXA          ;RECHTS/LINKS-FLAG
220: CC04 4A              LSR A
230: CC05 08              PHP
280: CC06 20 F1 B7        JSR CHKGET
290: CC09 E0 19           CPX #25     ;OBERE ZEILE
300: CC0B 90 02           BCC S1
310: CC0D A2 18           LDX #24
320: CC0F 8E F6 07 S1     STX OB
330: CC12 20 F1 B7        JSR CHKGET  ;UNTERE ZEILE
340: CC15 E0 19           CPX #25
350: CC17 90 02           BCC S2
360: CC19 A2 18           LDX #24
370: CC1B 8E F7 07 S2     STX UN
380: CC1E 8A              TXA
390: CC1F AE F6 07        LDX OB
400: CC22 AC F7 07        LDY UN
410: CC25 38              SEC
420: CC26 ED F6 07        SBC OB      ;OBEN - UNTEN
430: CC29 B0 08           BCS S3
440: CC2B 49 FF           EOR #$FF    ;OBEN<UNTEN=>TAUSCH
450: CC2D AE F7 07        LDX UN
460: CC30 AC F6 07        LDY OB
470: CC33 85 FE    S3     STA ZAHL    ;ZAEHLER
480: CC35 28              PLP
490: CC36 08              PHP         ;RE/LI-FLAG
500: CC37 90 03           BCC S4
510: CC39 C8              INY         ;RECHTS:ZEILE WEITER
520: CC3A 98              TYA         ;UND UNTEN STARTEN
530: CC3B AA              TAX
540: CC3C BD CB CC S4     LDA MULH,X  ;HIGH BYTE ADRESSE
```

```
550: CC3F 85 62          STA ADRESS+1
560: CC41 BD E5 CC       LDA MULL,X  ;LOW-BYTE
570: CC44 85 61          STA ADRESS
580: CC46 28             PLP
590: CC47 08             PHP         ;RE/LI-FLAG
600: CC48 90 08          BCC MOVE
610: CC4A E9 01          SBC #1      ;BEI RECHTS  -1
620: CC4C 85 61          STA ADRESS
630: CC4E B0 02          BCS MOVE
640: CC50 C6 62          DEC ADRESS+1
650:                ;
660:                ;VERSCHIEBUNG
670:                ;*************
680:                ;
690: CC52 A5 62     MOVE LDA ADRESS+1
700: CC54 29 03          AND #3
710: CC56 09 04          ORA #4      ;BASISADRESSE=$0400
720: CC58 28             PLP
730: CC59 08             PHP         ;FLAG
750: CC5A 20 86 CC       JSR MOVE1   ;VIDEORAM VERSCHIEBEN
760: CC5D 28             PLP
770: CC5E 08             PHP         ;FLAG
780: CC5F A5 61          LDA ADRESS
790: CC61 90 0A          BCC M1
800: CC63 69 27          ADC #39     ;C=1! / RECHTS
810: CC65 85 61          STA ADRESS
820: CC67 90 0C          BCC M2
830: CC69 E6 62          INC ADRESS+1
840: CC6B B0 08          BCS M2      ;UNBEDINGT
850: CC6D E9 27     M1   SBC #39     ;C=0! / LINKS
860: CC6F 85 61          STA ADRESS
870: CC71 B0 02          BCS M2
880: CC73 C6 62          DEC ADRESS+1
890: CC75 A5 62     M2   LDA ADRESS+1
900: CC77 29 03          AND #3
910: CC79 09 D8          ORA #$D8    ;FARBRAM BEI $D800
912: CC7B 28             PLP
915: CC7C 08             PHP
920: CC7D 20 86 CC       JSR MOVE1   ;FARBRAM VERSCHIEBEN
930: CC80 C6 FE          DEC ZAHL
940: CC82 10 CE          BPL MOVE
950: CC84 28             PLP
960: CC85 60             RTS
970:                ;
```

```
980:                    ;VERTEILER
990:                    ;**********
1000:                   ;
1010: CC86 85 62    MOVE1  STA ADRESS+1
1020: CC88 90 03           BCC LINKS
1030: CC8A 4C AB CC        JMP RECHTS
1040:                   ;
1050:                   ;LINKSVERSCHIEBUNG
1060:                   ;******************
1070:                   ;
1080: CC8D A0 00    LINKS  LDY #0
1090: CC8F B1 61           LDA (ADRESS),Y
1100: CC91 AA              TAX          ;ERSTES BYTE MERKEN
1140: CC92 A0 27           LDY #39
1150: CC94 B1 61    L2     LDA (ADRESS),Y
1160: CC96 48              PHA          ;MERKER 1
1170: CC97 8A              TXA          ;HOLE MERKER 2
1180: CC98 91 61           STA (ADRESS),Y
1190: CC9A 68              PLA          ;HOLE MERKER 1
1200: CC9B AA              TAX          ;IN MERKER 2
1210: CC9C 88              DEY
1220: CC9D 10 F5           BPL L2
1230: CC9F 18              CLC
1240: CCA0 A5 61           LDA ADRESS
1250: CCA2 69 28           ADC #40      ;NAECHSTE ZEILE
1260: CCA4 85 61           STA ADRESS
1270: CCA6 90 02           BCC L3
1280: CCA8 E6 62           INC ADRESS+1
1290: CCAA 60       L3     RTS
1300:                   ;
1310:                   ;RECHTSVERSCHIEBUNG
1320:                   ;*******************
1330:                   ;
1340: CCAB 38       RECHTS SEC
1350: CCAC A5 61           LDA ADRESS
1360: CCAE E9 28           SBC #40
1370: CCB0 85 61           STA ADRESS  ;40 ABZIEHEN
1380: CCB2 B0 02           BCS R1
1390: CCB4 C6 62           DEC ADRESS+1
1400: CCB6 A0 28    R1     LDY #40
1410: CCB8 B1 61           LDA (ADRESS),Y ;LINKES BYTE HOLEN
1420: CCBA AA              TAX
1460: CCBB A0 01           LDY #1
```

```
1470: CCBD B1 61    R3    LDA (ADRESS),Y
1480: CCBF 48             PHA         ;MERKER 1
1490: CCC0 8A             TXA         ;MERKER 2 HOLEN
1500: CCC1 91 61          STA (ADRESS),Y
1510: CCC3 68             PLA         ;MERKER 1
1520: CCC4 AA             TAX         ;IN MERKER 2
1530: CCC5 C8             INY
1540: CCC6 C0 29          CPY #41
1550: CCC8 D0 F3          BNE R3
1560: CCCA 60             RTS
1570: CCCB 04 04 04 MULH  .BYT 4,4,4,4,4,4,4,5,5,5,5,5,5
1580: CCD8 06 06 06       .BYT 6,6,6,6,6,6,6,7,7,7,7,7,7
1590: CCE5 00 28 50 MULL  .BYT $00,$28,$50,$78,$A0,$C8,$F0
1600: CCEC 18 40 68       .BYT $18,$40,$68,$90,$B8,$E0
1610: CCF2 08 30 58       .BYT $08,$30,$58,$80,$A8,$D0,$F8
1620: CCF9 20 48 70       .BYT $20,$48,$70,$98,$C0,$E8
```

Auch hier wird Ihnen natürlich wieder ein entsprechender BASIC-Lader angeboten:

```
100 FOR I = 52224 TO 52480
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA  32,241,183,138, 74,  8, 32,241,183,224, 25,144
130 DATA   2,162, 24,142,246,  7, 32,241,183,224, 25,144
140 DATA   2,162, 24,142,247,  7,138,174,246,  7,172,247
150 DATA   7, 56,237,246,  7,176,  8, 73,255,174,247,  7
160 DATA 172,246,  7,133,254, 40,  8,144,  3,200,152,170
170 DATA 189,203,204,133, 98,189,229,204,133, 97, 40,  8
180 DATA 144,  8,233,  1,133, 97,176,  2,198, 98,165, 98
190 DATA  41,  3,  9,  4, 40,  8, 32,134,204, 40,  8,165
200 DATA  97,144, 10,105, 39,133, 97,144, 12,230, 98,176
210 DATA   8,233, 39,133, 97,176,  2,198, 98,165, 98, 41
220 DATA   3,  9,216, 40,  8, 32,134,204,198,254, 16,206
230 DATA  40, 96,133, 98,144,  3, 76,171,204,160,  0,177
240 DATA  97,170,160, 39,177, 97, 72,138,145, 97,104,170
250 DATA 136, 16,245, 24,165, 97,105, 40,133, 97,144,  2
260 DATA 230, 98, 96, 56,165, 97,233, 40,133, 97,176,  2
270 DATA 198, 98,160, 40,177, 97,170,160,  1,177, 97, 72
280 DATA 138,145, 97,104,170,200,192, 41,208,243, 96,  4
290 DATA   4,  4,  4,  4,  4,  4,  5,  5,  5,  5,  5,  5
300 DATA   6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,  7
310 DATA   7,  0, 40, 80,120,160,200,240, 24, 64,104,144
320 DATA 184,224,  8, 48, 88,128,168,208,248, 32, 72,112
330 DATA 152,192,232,  0,  0
340 IF S <> 27098 THEN PRINT "FEHLER IN DATAS !!" : END
350 PRINT "OK"
```

Dieses Programm harmonisiert mit dem Graphik-Paket aus Kapitel 5, d.h. beide Maschinenprogramme können sich gleichzeitig im Speicher befinden und auch verwendet werden. Der Aufruf erfolgt, wie gesagt, ebenfalls über SYS und zwar in der wie folgt angegebenen Art und Weise:

```
SYS 52224,r,a,e
```

Dabei bedeuten:

r: Richtung des Scrollens (0=links/1=rechts)
a: Anfangszeile und
e: Endzeile zwischen denen gescrollt wird

Die Anwendung dieser BASIC-Erweiterung sollten Sie dem folgenden kleinen Demonstrationsprogramm entnehmen:

```
100 REM ******************
110 REM **              **
120 REM **  SCROLLING   **
130 REM **              **
140 REM ******************
150 REM
200 SR = 52224 : REM SCROLL-ADRESSE
210 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
220 PRINT:PRINT"  DER SCROLL-BEFEHL ERMOEGLICHT IHNEN,"
230 PRINT"     JEDE BELIEBIGE BILDSCHIRMZEILE"
240 PRINT"  UND BEI BEDARF AUCH MEHRERE GLEICH-"
250 PRINT"        ZEITIG ZU VERSCHIEBEN."
260 PRINT"              SEHEN SIE?"
270 FOR X=1 TO 5000 : NEXT X
280 FOR X=1 TO 40
290 FOR Y=1 TO 50 : NEXT Y
300 SYS SR,1,6,6
310 NEXT X
320 PRINT:PRINT "DAS GANZE GEHT NATUERLICH AUCH SCHNELLER"
330 FOR X=1 TO 4000 : NEXT X
340 FOR X=1 TO 200 : SYS SR,1,8,8 : NEXT X
350 PRINT"          UND ANDERS HERUM!"
360 FOR X=1 TO 4000 : NEXT X
370 FOR X=1 TO 400 : SYS SR,0,10,10 : NEXT X
380 PRINT : PRINT"  VIELLEICHT AUCH DER GANZE BILDSCHIRM"
390 FOR X=1 TO 4000 : NEXT X
400 FOR X=1 TO 200 : SYS SR,0,0,24 : NEXT X
410 PRINT:PRINT"           WOLLEN SIE EINMAL"
```

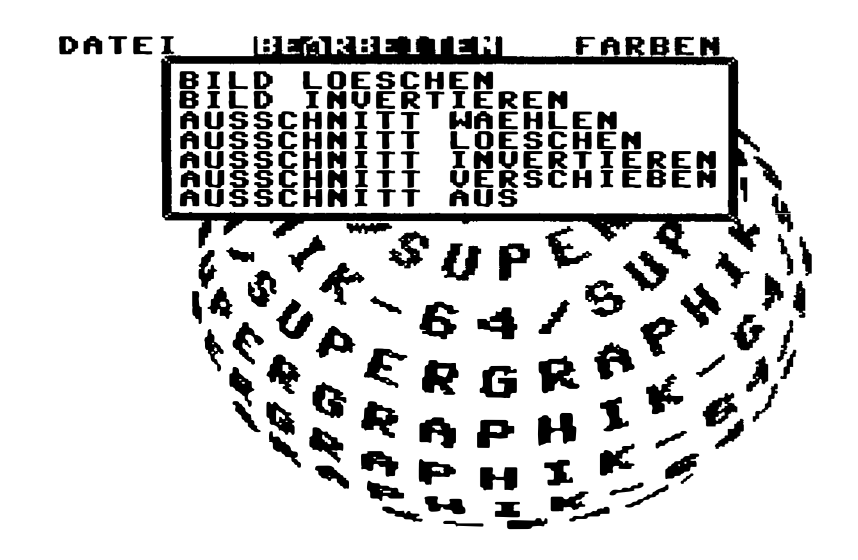
```
420 PRINT"        LAUFSCHRIFTEN ERSTELLEN?"
430 FOR X=1 TO 4000 : NEXT X
440 PRINT:PRINT"         ---- DATA BECKER ----"
450 FOR X=1 TO 400:SYS SR,0,17,17:FOR Y=X TO 150: NEXT Y,X
```

Wenn Sie mit unserer Supergraphik arbeiten, wird Ihnen dieser Befehl in erweiterter Form bereits durch GMOVE angeboten.

Wir haben Ihnen nun die wichtigsten Grundlagen für die Programmierung der Spiele vermittelt. Nun ist es an Ihnen, diese in die Tat umzusetzen. Die Phantasie können wir Ihnen nicht abnehmen. Und wenn Sie einmal ein kleines Spiel fertig haben, dann laden Sie mich doch einmal ein.

3.4 GEM 64 alias GEOS - Bildschirmfenster mit Supergraphik

Zusammen mit dem Commodore 64 wird seit neuestem das Bildschirmfenster-orientierte GEOS ausgeliefert. Es simuliert das bekannte GEM vom Atari ST oder anderen Rechnern auf dem kleinen 64er. Daß das auch mit der Supergraphik und sogar in BASIC funktioniert und auch gar nicht so unheimlich schwer ist, soll ihnen das unten stehende Programm zeigen. Dort wurde ein kleines GEM-ähnliches "Betriebssystem" entworfen, das schon recht nett ist und sich sehen lassen kann. Es wurde absichtlich so entwickelt, daß auch Sie sehr leicht Änderungen und Erweiterungen vornehmen können. Doch schauen Sie einmal selbst:



```
100 REM ***************************
110 REM **                       **
120 REM **  BILDSCHIRMFENSTER    **
130 REM **                       **
140 REM ***************************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0 : SCOL= 1,7
165 GMODE 1,1 : GCLEAR : SCOL= 0,0 : SCOL= 1,7
170 MH=9 : MZ= 3     :REM MENUEFARBEN
180 KH=2 : KZ= 7     :REM FENSTERFARBEN
190 GMODE 0
200 GOSUB 1050       :REM INIT/MENUELEISTE
210 REM
220 GOSUB 1650       :REM PFEIL STELLEN
230 IF KN=0 THEN 220 :REM KEIN KNOPF GEDRUECKT
240 REM
250 REM KNOPF GEDRUECKT:
260 REM
270 IF YP>7 THEN 220 :REM NICHT AUF MENUELEISTE
280 REM FESTSTELLEN DES MENUEPUNKTES:
290 FOR MP=0 TO AM
300  IF  XP<L1(MP)  OR  XP>L2(MP)  THEN  NEXT  MP  :  GOTO  220  :REM  NICHT
GEFUNDEN
310 REM
320 REM MENUEPUNKT ANGEWAEHLT:
330 REM
340 GOSUB 2300       :REM FENSTER OEFFNEN
350 GOSUB 1650       :REM PFEIL STELLEN
360 IF KN=0 THEN 350 :REM KEIN KNOPF GEDRUECKT
370 FOR UM=0 TO MZ(MP)
380 IF XP<X1(MP) OR YP<Y1(MP)+UM*8+ 5 THEN NEXT UM : GOSUB 2550 : GOTO
220
390 IF XP>X2(MP) OR YP>Y1(MP)+UM*8+13 THEN NEXT UM : GOSUB 2550 : GOTO
220
400 REM
410 REM UNTERMENUEPUNKT ANGEWAEHLT:
420 REM
430 GOSUB 2550   :REM MENUEFENSTER SCHLIESSEN
440 REM
450 REM UNTERMENUEPUNKT AUSFUEHREN:
460 REM
470 ON US(MP,UM) GOSUB 5080,5250,5650,5850,6050
480     IF     US(MP,UM)>5     THEN     ON     US(MP,UM)-5     GOSUB
6250,6450,6650,6950,7150,7350,7750
490 IF US(MP,UM)>12 THEN ON US(MP,UM)-12 GOSUB 8050,8450,8250
500 GOTO 220
1000 REM
1010 REM
```

```
1020 REM MENUELEISTE AUFBAUEN/INIT:
1030 REM ***************************
1040 REM
1050 RESTORE      :REM DATA-START
1060 READ AM      :REM ANZAHL MENUEPUNKTE
1070 AM=AM-1
1080 DIM L1(AM),L2(AM),MP$(AM),MZ(AM)
1090 DIM X1(AM),Y1(AM),X2(AM),Y2(AM)
1100 DIM UP$(AM,10),US(AM,10)
1110 REM
1120 REM MENUEPUNKT-SCHLEIFE:
1130 REM (DATAS EINLESEN)
1140 FOR MP=0 TO AM
1150 READ L1(MP),L2(MP) :REM MENUEPUNKTPOSITION
1160 READ MP$(MP) :REM MENUEPUNKT
1170 READ MZ(MP)  :REM ANZAHL UNTERMENUES
1180 MZ(MP)=MZ(MP)-1
1190 READ X1(MP),Y1(MP),X2(MP),Y2(MP) :REM KOORD.
1200 REM
1210 REM UNTERMENUEPUNKT-SCHLEIFE:
1220 REM
1230 FOR UP=0 TO MZ(MP)
1240 READ UP$(MP,UP) :REM UNTERMENUEPUNKTE
1250 READ US(MP,UP)  :REM SPRUNGADRESSEN
1260 NEXT UP
1270 NEXT MP
1280 REM
1290 REM MENUELEISTE ZEICHNEN:
1300 REM
1310 PCOL= 0,MH : PCOL= 1,MZ
1320 GCLEAR ,0,0 TO 319,7      :REM STREIFEN
1330 FOR MP=0 TO AM
1340 TEXT 2,MP$(MP),L1(MP),7,0 :REM SCHREIBEN
1350 NEXT MP
1360 REM
1370 REM WEITERE INITIALISIERUNG:
1380 REM
1390 XP=160 : YP=100  :REM PFEILPOSITION
1400 G=1
1410 PF$=".2.2.2.2.10.0.0.0.1.1.1.23.3.2.12.12.1" :REM PFEILDEFINTION
1420 SCALE= 0,0
1430 GOSUB 2080
1440 RETURN
1600 REM
1610 REM
```

```
1620 REM PFEILSTEUERUNG:
1630 REM ****************
1640 REM
1650 KN=0        :REM KNOPFFLAG LOESCHEN
1660 GET T$ : T=ASC(T$+CHR$(0)) :REM TASTE HOLEN
1670 IF# B 2 THEN KN=1     :REM KNOPF
1680 IF# U 2 THEN YP=YP-G  :REM HOCH
1690 IF# D 2 THEN YP=YP+G  :REM RUNTER
1700 IF# R 2 THEN XP=XP+G  :REM RECHTS
1710 IF# L 2 THEN XP=XP-G  :REM LINKS
1720 IF T=133 THEN KN=1    :REM KNOPF
1730 IF T=145 THEN YP=YP-G :REM HOCH
1740 IF T= 17 THEN YP=YP+G :REM RUNTER
1750 IF T= 29 THEN XP=XP+G :REM RECHTS
1760 IF T=157 THEN XP=XP-G :REM LINKS
1770 IF T= 20 THEN G=G+7 : IF G>8 THEN G=1
1780 IF XP<  0 THEN XP=XP+319
1790 IF YP<  0 THEN YP=YP+199
1800 IF XP>319 THEN XP=0
1810 IF YP>199 THEN YP=0
1820 IF X0=XP AND Y0=YP THEN 1870 :REM KEINE VERAENDERUNG
1840 GOSUB 2080    :REM PFEIL LOESCHEN
1850 X0=XP : Y0=YP :REM PFEILKOORDINATEN
1860 GOSUB 2080    :REM PFEIL SETZEN
1870 RETURN
2000 REM
2010 REM
2020 REM PFEIL SETZEN/LOESCHEN:
2030 REM ***********************
2040 REM
2050 REM UEBERGABE:
2060 REM            X0,Y0:KOORDINATEN
2070 REM
2080 DRAW 2,PF$ ON X0,Y0 :REM PFEIL SETZEN
2090 RETURN
2200 REM
2210 REM
2220 REM UNTERMENUEFENSTER ZEICHNEN:
2230 REM ****************************
2240 REM
2250 REM UEBERGABE:
2260 REM            MP - MENUEPUNKT
2270 REM
2300 GOSUB 2080 :REM PFEIL LOESCHEN
2310 PCOL= 0,MH : PCOL= 1,MZ
2320 INVERS 255,L1(MP),0 TO L2(MP),7 :REM INVERTIERE MENUEPUNKT
2330 GMODE 1    :REM ZIELSEITE 2
2340  GCOMB  3,X1(MP),Y1(MP)  TO  X2(MP),Y2(MP)  :REM  INHALT  DES  FENSTERS
RETTEN
2350 GMODE 0    :REM WIEDER AUF 1
2360 GCLEAR ,X1(MP),Y1(MP) TO X2(MP),Y2(MP) :REM FENSTER LOESCHEN
```

```
2370 FRAME 2,3,X1(MP),Y1(MP) TO X2(MP),Y2(MP) :REM RAHMEN
2380 REM
2390 REM
2400 FOR UM=0 TO MZ(MP)
2410  TEXT  ,UP$(MP,UM),X1(MP)+5,Y1(MP)+UM*8+13,0  :REM  UNTERMENUEPUNKT
AUSGEBEN
2420 NEXT UM
2430 GOSUB 2080 :REM PFEIL SETZEN
2440 RETURN
2500 REM
2510 REM
2520 REM MENUEFENSTER SCHLIESSEN:
2530 REM *************************
2540 REM
2550 GOSUB 2080 :REM PFEIL LOESCHEN
2560 PCOL= 0,MH : PCOL= 1,MZ
2570 INVERS 255,L1(MP),0 TO L2(MP),7 :REM NORMALISIERE MENUEPUNKT
2580 GCOMB 3,X1(MP),Y1(MP) TO X2(MP),Y2(MP) :REM ALTER INHALT AUS SEITE
2
2590 GOSUB 2080 :REM PFEIL SETZEN
2600 RETURN
2700 REM
2710 REM
2720 REM KOMMUNIKATIONSFENSTER:
2730 REM ***********************
2740 REM UEBERGABE:
2750 REM           S1$-AUSGABESTRING 1
2760 REM           S2$-AUSGABESTRING 2
2770 REM           ZA -ZAHL DER EINGABEZEICHEN
2775 REM AUSGABE:  BK$-EINGABEZEILE
2780 REM
2790 PCOL= 0,KH : PCOL= 1,KZ
2800 GOSUB 2080  :REM PFEIL LOESCHEN
2810 GMODE 1     :REM GRAPHIKSEITE 2
2820 GCOMB 3,80,80 TO 239,119 :REM FENSTER RETTEN
2825 GMODE 0
2830 GCLEAR ,80,80 TO 239,119 :REM FENSTER LOESCHEN
2840 FRAME 2,3,80,80 TO 239,119 :REM RAHMEN
2850 TEXT ,S1$,90, 93,0       :REM TEXT 1 AUSGEBEN
2860 TEXT ,S2$,90,103,0       :REM TEXT 2
2870 REM
2880 Z=-1 : BK$=""
2890 Z=Z+1
2900 GET T$ : IF T$="" THEN 2900
2910 IF ASC(T$)=13 THEN 2960
2920 IF Z=ZA THEN 2900
2930 TEXT ,T$,90+8*Z,113,0 :REM ZEICHEN AUSGEBEN
2940 BK$=BK$+T$
2950 GOTO 2890
2960 REM
```

```
2970 REM FENSTER SCHLIESSEN:
2980 REM
2990 GCOMB 3,80,80 TO 239,119  :REM FENSTER WIEDERHOLEN
3000 GOSUB 2080  :REM PFEIL SETZEN
3010 RETURN
5000 REM
5010 REM *****************
5020 REM BEFEHLSROUTINEN:
5030 REM *****************
5040 REM
5050 REM PROGRAMM LADEN:
5060 REM ****************
5070 REM
5080 S1$="GEBEN SIE DEN"
5090 S2$="PROGRAMMNAMEN AN:"
5100 ZA =15
5110 GOSUB 2790 :REM EINGABE HOLEN
5120 IF BK$="" THEN 5140
5130 REM LOAD BK$,8 :REM PROGRAMM LADEN UND STARTEN
5140 RETURN
5200 REM
5210 REM
5220 REM BILD EINLADEN:
5230 REM ***************
5240 REM
5250 S1$="GEBEN SIE DEN"
5260 S2$="BILDNAMEN AN:"
5270 ZA=15
5280 GOSUB 2790      :REM EINGABE HOLEN
5285 IF BK$="" THEN 5450
5290 GMODE 2
5300 GLOAD GC,2,BK$,8 :REM EINLADEN (SEITE 2)
5310 GMODE 0
5320 GOSUB 2080       :REM PFEIL AUS
5330 IF AF=0 THEN 5400
5340 REM
5350 REM AUSSCHNITT NACH 2:
5360 REM
5370 GMODE 1      :REM SEITE 2=ZIEL
5380 GCOMB 3,XA,YA TO XB,YB :REM AUSSCHNITT NACH 2
5390 GMODE 0      :REM SEITE 1
5400 REM
5410 REM NACH SEITE 1 COPIEREN:
5420 REM
5430 GCOMB 3,0,8 TO 319,199
5440 GMODE 1 : GCLEAR : GMODE 0
5450 GOSUB 2080  :REM PFEIL EIN
5460 RETURN
5600 REM
5610 REM
```

```
5620 REM DATEI LOESCHEN:
5630 REM ****************
5640 REM
5650 S1$="GEBEN SIE DEN"
5660 S2$="DATEINAMEN AN:"
5670 ZA=15
5680 GOSUB 2790 :REM EINGABE HOLEN
5690 IF BK$="" THEN 5710
5700 OPEN 1,8,15,"S0:"+BK$
5710 RETURN
5800 REM
5810 REM
5820 REM DISKETTE FORMATIEREN:
5830 REM **********************
5840 REM
5850 S1$="DISKETTE LOESCHEN!"
5860 S2$="DISKNAME,ID:"
5870 ZA=18
5880 GOSUB 2790 :REM EINGABE HOLEN
5890 IF BK$="" THEN 5910
5900 OPEN 1,8,15,"N0:"+BK$
5910 RETURN
6000 REM
6010 REM
6020 REM BILD SPEICHERN:
6030 REM ****************
6040 REM
6050 S1$="GEBEN SIE DEN"
6060 S2$="BILDNAMEN AN:"
6070 ZA=15
6080 GOSUB 2790 :REM EINGABE HOLEN
6090 IF BK$="" THEN 6110
6100 GSAVE GC,1,BK$,8
6110 RETURN
6200 REM
6210 REM
6220 REM BILD LOESCHEN:
6230 REM ***************
6240 REM
6250 S1$="MOECHTEN SIE"
6260 S2$="LOESCHEN (J/N)?"
6270 ZA=2
6280 GOSUB 2790 :REM EINGABE HOLEN
6290 IF BK$<>"J" THEN GOTO 6390
6300 GOSUB 2080 :REM PFEIL AUS
6310 IF AF=0 THEN:GCLEAR ,0,8 TO 319,199 : GOTO 6380
6320 GMODE 1     :REM SEITE 2=ZIEL
6330 GCOMB 3,XA,YA TO XB,YB :REM FENSTER RETTEN
6340 GMODE 0     :REM SEITE 1
6350 GCLEAR ,0,8 TO 319,199 :REM LOESCHEN
6360 GCOMB 3,XA,YA TO XB,YB :REM FENSTER ZURUECK
```

```
6370 REM
6380 GOSUB 2080 :REM PFEIL SETZEN
6390 RETURN
6400 REM
6410 REM
6420 REM BILD INVERTIEREN:
6430 REM *****************
6440 REM
6450 GOSUB 2080 :REM PFEIL AUS
6460 IF AF=0 THEN:INVERS ,0,8 TO 319,199 : GOTO 6530
6470 GMODE 1     :REM SEITE 2=ZIEL
6480 GCOMB 3,XA,YA TO XB,YB :REM FENSTER RETTEN
6490 GMODE 0     :REM SEITE 1
6500 INVERS ,0,8 TO 319,199 :REM INVERTIEREN
6510 GCOMB 3,XA,YA TO XB,YB :REM FENSTER ZURUECK
6520 REM
6530 GOSUB 2080 :REM PFEIL EIN
6540 RETURN
6600 REM
6610 REM
6620 REM AUSSCHNITT WAEHLEN:
6630 REM ********************
6640 REM
6650 S1$="CURSOR AUF START"
6660 S2$="  UND KNOPF!"
6670 ZA=0
6680 GOSUB 2790 :REM MELDUNG GEBEN
6690 GOSUB 1650 :REM PFEIL STELLEN
6700 IF KN=0 OR YP<8 THEN 6690
6710 REM
6720 XA=XP : YA=YP :REM KOORDINATEN UEBERNEHMEN
6730 FRAME 2,1,XA,YA TO XP,YP :REM RAHMEN ZEICHNEN
6740 FRAME 2,1,XA,YA TO XP,YP :REM RAHMEN LOESCHEN
6750 GOSUB 1650 :REM PFEIL STELLEN
6760 IF KN=0 OR YP<8 THEN 6730
6770 XB=XP : YB=YP :REM KOORDINATEN UEBERNEHMEN
6780 AF=1 :REM FLAG
6790 RETURN
6900 REM
6910 REM
6920 REM AUSSCHNITT LOESCHEN:
6930 REM *********************
6940 REM
6950 IF AF=0 THEN 7040
6960 S1$="MOECHTEN SIE"
6970 S2$="LOESCHEN (J/N)?"
6980 ZA=2
6990 GOSUB 2790 :REM EINGABE HOLEN
7000 IF BK$<>"J" THEN 7040
7010 GOSUB 2080 :REM PFEIL AUS
7020 GCLEAR ,XA,YA TO XB,YB
```

```
7030 GOSUB 2080 :REM PFEIL EIN
7040 RETURN
7100 REM
7110 REM
7120 REM AUSSCHNITT INVERTIEREN:
7130 REM ************************
7140 REM
7150 IF AF=0 THEN 7200
7160 GOSUB 2080 :REM PFEIL AUS
7170 INVERS ,XA,YA TO XB,YB
7180 GOSUB 2080 :REM PFEIL EIN
7190 RETURN
7300 REM
7310 REM
7320 REM AUSSCHNITT VERSCHIEBEN:
7330 REM ************************
7340 REM
7350 IF AF=0 THEN 7620
7360 FRAME 2,1,XP,YP TO XP+XB-XA,YP+YB-YA :REM RAHMEN ZEICHNEN
7370 FRAME 2,1,XP,YP TO XP+XB-XA,YP+YB-YA :REM RAHMEN LOESCHEN
7380 GOSUB 1650 :REM PFEIL STELLEN
7390 IF KN=0 OR YP<8 THEN 7360
7400 REM
7410 REM VERSCHIEBEN:
7420 REM
7430 GOSUB 2080 :REM PFEIL AUS
7440 GMODE 1
7450 GCOMB 3,XA,YA TO XB,YB TO XP,YP
7550 GMODE 0
7560 GMODE 0
7570 XB=XP+XB-XA : YB=YP+YB-YA :REM KOORDINATEN UEBERNEHMEN
7580 XA=XP : YA=YP
7590 GCOMB 3,XA,YA TO XB,YB    :REM NACH SEITE 1
7600 GOSUB 2080 :REM PFEIL EIN
7610 REM
7620 RETURN
7700 REM
7710 REM
7720 REM AUSSCHNITT AUS:
7730 REM ***************
7740 REM
7750 S1$="MOECHTEN SIE"
7760 S2$="AUSSCHALTEN (J/N)?"
7770 ZA=2
7780 GOSUB 2790 :REM EINGABE HOLEN
7790 IF BK$<>"J" THEN 7810
7800 AF=0 :REM FLAG AUS
7810 RETURN
```

```
8000 REM
8010 REM
8020 REM HINTERGRUNDFARBE:
8030 REM ******************
8040 REM
8050 S1$="GEBEN SIE DIE"
8060 S2$="HINTERGRUNDF. AN:"
8070 ZA=2
8080 GOSUB 2790 :REM EINGABE HOLEN
8090 IF BK$="" THEN 8170
8100 GOSUB 2080 :REM PFEIL AUS
8110 GMODE 1
8120 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE RETTEN
8130 GMODE 0
8140 SCOL= 0,VAL(BK$):H1=VAL(BK$)
8150 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE HOLEN
8160 GOSUB 2080 :REM PFEIL EIN
8170 RETURN
8200 REM
8210 REM
8220 REM ZEICHENFARBE:
8230 REM **************
8240 REM
8250 S1$="GEBEN SIE DIE"
8260 S2$="ZEICHENFARBE AN:"
8270 ZA=2
8280 GOSUB 2790 :REM EINGABE HOLEN
8290 IF BK$="" THEN 8370
8300 GOSUB 2080 :REM PFEIL AUS
8310 GMODE 1
8320 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE RETTEN
8330 GMODE 0
8340 SCOL= 1,VAL(BK$)
8350 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE HOLEN
8360 GOSUB 2080 :REM PFEIL EIN
8370 RETURN
8400 REM
8410 REM
8420 REM RAHMENFARBE:
8430 REM *************
8440 REM
8450 S1$="GEBEN SIE DIE"
8460 S2$="RAHMENFARBE AN:"
8470 ZA=2
8480 GOSUB 2790 :REM EINGABE HOLEN
8490 IF BK$="" THEN 8570
8500 GOSUB 2080 :REM PFEIL AUS
8510 GMODE 1
8520 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE RETTEN
8530 GMODE 0
8540 COLOR= VAL(BK$),H1
```

```
8550 GCOMB 3,0,0 TO 319,7 :REM MENUELEISTE HOLEN
8560 GOSUB 2080 :REM PFEIL EIN
8570 RETURN
10000 REM
10010 REM
10020 REM DATEN FUER BILDSCHIRMFENSTER:
10030 REM ******************************
10040 REM
10050 REM MENUEZEILE:
10060 REM
10070 DATA 3 :REM ANZAHL MENUEPUNKTE
10080 REM
10090 DATA 16,56 :REM POSITION MENUEPUNKT 1
10100 DATA DATEI :REM MENUEPUNKT
10110 REM
10120 DATA 5     :REM ZAHL DER UNTERMENUES
10130 DATA  10, 8,132, 60   :REM BILDSCHIRMFENSTER
10140 DATA PROGRAMM LADEN,1 :REM UNTERMENUES/SPRUNGADRESSEN:
10150 DATA BILD LADEN,2
10160 DATA DATEI LOESCHEN,3
10170 DATA FORMATIEREN,4
10180 DATA BILD SPEICHERN,5
10190 REM
10200 REM
10210 DATA 80,160
10220 DATA BEARBEITEN
10230 REM
10240 DATA 7
10250 DATA  50,  8,236, 74
10260 DATA BILD LOESCHEN,6
10270 DATA BILD INVERTIEREN,7
10280 DATA AUSSCHNITT WAEHLEN,8
10290 DATA AUSSCHNITT LOESCHEN,9
10300 DATA AUSSCHNITT INVERTIEREN,10
10310 DATA AUSSCHNITT VERSCHIEBEN,11
10320 DATA AUSSCHNITT AUS,12
10330 REM
10340 REM
10350 DATA 184,232
10360 DATA FARBEN
10370 REM
10380 DATA 3
10390 DATA 140,  8,246, 42
10400 DATA HINTERGRUND,13
10410 DATA RAHMEN,14
10420 DATA ZEICHENFARBE,15
```

Haben Sie keine Angst vor der Länge dieses Programmes, wir werden es Schritt für Schritt erläutern.

Die eigentliche Hauptroutine füllt lediglich die Zeilen 160-500 aus, der Rest sind Unterprogramme und Tabellen. Was wird gemacht? Nachdem in den Zeilen 160/165 die beiden Graphikseiten gelöscht wurden, werden zunächst einige Initialisierungen vorgenommen. Diese Initialisierungen finden im Unterprogramm ab Zeile 1050 statt und betreffen die Datenübernahme aus den DATA-Zeilen ab Zeile 10000 und die Ausgabe der oberen Menueleiste.

Die obere Menueleiste enthält eine bestimmte Anzahl von Menuepunkten (hier 3), die durch das READ AM der Zeile 1070 eingelesen wird. Jeder dieser Menuepunkte soll später angewählt werden und eine Reihe von Untermenuepunkten freigeben, die ebenfalls bereits ab Zeile 10000 definiert sind.

Dazu werden zunächst einmal eine ganze Menge Dimensionierungen vorgenommen (Zeilen 1080-1100). Die einzelnen Speicher haben folgende Funktion:

L1() - Startposition des Menuepunktes
L2() - Endposition des Menuepunktes
MP$() - Name des Menuepunktes
MZ() - Anzahl der Untermenuepunkte
X1() - x-Koord. oben links des Pull-down-Menues
Y1() - y-Koord. oben links des Pull-down-Menues
X2() - x-Koord. unten rechts des Pull-down-Menues
Y2() - y-Koord. unten rechts des Pull-down-Menues
UP$(,) - Name des Untermenuepunktes
US(,) - Startadressennummer des Untermenuepunktes

In den Zeilen 1140-1270 werden diese Daten aus den DATAs der Zeilen 10000 ff. ausgelesen. Dann kann mit einem Teil dieser Daten die Menueleiste in den Zeilen 1310-1350 gezeichnet werden.

Die weiteren Initialisierungen betreffen die Pfeil-Start-Position, die Pfeildefinition für den DRAW-Befehl, die SCALE=-Definition des Pfeiles und das Einschalten desselben durch das Unterprogramm ab Zeile 2080.

In diesem Unterprogramm wird der Pfeil durch einen DRAW-Befehl invertiert gezeichnet, d.h. bei einem nochmaligen Aufruf dieser Routine wird er wieder gelöscht. Dieser Vorgang wird uns noch durch das gesamte Programm begleiten.

Doch kommen wir zurück zu unserem Hauptteil: Nun beginnt die eigentliche Hauptschleife des Programms (Z. 220). Hier wird zunächst einmal eine Routine (ab Z. 1650) aufgerufen, die auf die Cursortasten bzw. die Joystick-Steuerung anspricht, um einen kleinen Pfeil über den Bildschirm zu bewegen. Mit diesem Pfeil können Sie die einzelnen Menuepunkte anwählen etc. Wenn Sie den Feuerknopf Ihres Joysticks oder die f1-Taste betätigen, dann wird die Variable KN (für Knopf) auf 1 gesetzt. Dies ist ein Zeichen für die Übergeordnete Routine und wird ihr übergeben. Mit der Taste <inst/del> können Sie die Geschwindigkeit des Pfeiles erhöhen.

Auf dieses KN wird dann auch in Zeile 230 getestet. Solange kein Knopf gedrückt ist, können Sie nur den Pfeil bewegen.

Ist nun aber KN=1, dann wird die Position des Pfeiles überprüft. Um einen Menuepunkt anzuwählen, muß sich die Spitze des Pfeiles innerhalb der Menuezeile und innerhalb der Start- und Endpositionen eines Menuepunktes befinden (Z. 270-300).

Ist das der Fall, so ist ein Menuepunkt angewählt und es wird durch das Unterprogramm ab Zeile 2300 ein Pull-down-Menue heruntergelassen. Diesem Unterprogramm wird lediglich mit MP die Nummer des Menuepunktes übergeben, dessen Untermenues angezeigt werden sollen.

Jetzt wird es interessant. Nachdem zunächst einmal der Pfeil gelöscht wurde (Z. 2300) wird der Menuepunkt invertiert (Z. 2320). Dann muß der Inhalt des Graphikbildschirms, der nachher vom Pull-down-Menue überdeckt werden soll, in die zweite Graphikseite gerettet werden. Dies geschieht mit Hilfe des GCOMB-Befehles in Zeile 2340, der diesen Bereich in die zweite Graphikseite kopiert. Alsdann wird der Bereich in Seite 1 gelöscht (Z. 2360) und nach dem Zeichnen eines Rahmens die verschiedenen Untermenuepunkte eingeschrieben. Die Defintion des Fensters und der Untermenuepunkte stammen dabei bekanntlich aus den DATAs ab Zeile 10000.

Das Pull-down-Menue mit den verschiedenen Untermenuepunkten ist nun also sichtbar. Jetzt kann die Kontrolle wieder dem Benutzer übergeben werden, der den Pfeil auf den Punkt ansteuert, den er auswählen möchte. Welchen er angeklickt hat, wird dann in den beiden Zeilen 380/390 festgestellt. Wurde keiner angewählt, so wird das Pull-down-Menue wieder geschlossen und zum Programmanfang gesprungen.

Das Schließen des Pull-down-Menues findet ab der Zeile 2550 statt. Das geht einfach durch Zurückkopieren des geretteten Graphikteils und dem Normalisieren des Menuepunktes.

Wurde nun tatsächlich ein Untermenuepunkt angewählt, so wird das Pull-down-Menue ebenfalls geschlossen (Z. 430). Sie haben sich sicher bereits gefragt, was wir unter Sprungadressennummer des Speichers US(,) verstehen. Nun, das ist hier ganz einfach. Jedem Untermenuepunkt ist eine ganz bestimmte Nummer zugeordnet, die in den DATAs direkt hinter dem Namen des Untermenuepunktes steht. Sie bestimmt in den Zeilen 470-490 die Zeilennummer, zu der ein Unterprogrammsprung ausgeführt werden soll. In diesem Unterprogramm wird dann ausgeführt, was Sie angewählt haben. Danach springt das Programm wieder nach Zeile 220.

Was nun in den einzelnen Untermenues passiert, soll hier nicht mehr erläutert werden, da es sich praktisch nur noch um den Aufruf verschiedener Unterprogramme und Lösch-, Invertier- und Kopierbefehle handelt, die leicht zu durchschauen sein sollten. Beispielhaft werden wir uns hier nur einmal den Untermenuepunkt "Ausschnitt wählen" anschauen (ab Zeile 6650):

Sie können einen bestimmten rechteckigen Bildausschnitt anwählen, den Sie getrennt verschieben, invertieren oder löschen können oder der beim Löschen oder Invertieren des Graphikbildes vom Löschen bzw. Invertieren ausgenommen sein soll, d.h. das gesamte Bild wird gelöscht, invertiert oder von Diskette nachgeladen, bis auf den definierten Ausschnitt.

Dieser Ausschnitt wird nun in dem Unterprogramm ab Zeile 6650 festgelegt. Zunächst aber sollten wir ein kleines Unterprogramm vorstellen, das wir bisher noch nicht kannten. Es ist das Unterprogramm ab Zeile 2790, das eine Eingabe vom Benutzer einholt.

Dieser Routine werden in den Speichern S1$ und S2$ zwei Strings übergeben, deren Funktion Sie gleich erkennen werden. Weiterhin übergeben Sie ihr die Variable ZA, die die Anzahl der Zeichen bestimmt, die der Benutzer maximal eingeben kann. Die Eingabe wird dann durch den Speicher BK$ an die übergeordnete Routine übergeben.

Dieses Unterprogramm öffnet zunächsteinmal ein Kommunikationsfenster in der Mitte des Bildschirms auf die gleiche Art und Weise wie wir das Pull-down-Menue erstellt haben, indem zunächst der Inhalt des betroffenen Bereiches in die zweite Seite gerettet wird. Der Bereich wird dann gelöscht und umrahmt. In diesem Stadium werden die beiden übergebenden Strings in den Zeilen 2850/2860 in dieses Bildschirmfenster geschrieben. Sie enthalten eine Meldung bzw. die Aufforderung, eine Eingabe zu tätigen.

Diese Eingabe wird dann in den Zeilen 2890-2950 getätigt. Jede Eingabe wird mit <return> abgeschlossen. Ist ZA=0, so brauchen Sie nur <return> einzugeben, um eine Meldung z.B. zu bestätigen.

Nach der Eingabe, die nicht editiert werden kann (eine Aufgabe für Sie!), wird das Fenster durch Zurückholen des ursprünglichen Inhaltes (Zeile 2990) wieder geschlossen.

Kommen wir zurück zur Zeile 6650: Ausschnitt wählen. In diesem Fall wird der eben vorgestellten Routine nur eine Meldung übergeben, die mit <return> abgeschlossen werden muß. Dann können Sie den Pfeil wieder frei bewegen. Beim nächsten Knopfdruck (oder f1) jedoch legen Sie die obere linke Ecke des Ausschnittes fest. Danach erscheint mit der Pfeilbewegung ein blinkendes Rechteck, das diesen Ausschnitt kennzeichnet (Zeilen 6730/6749). Bei nochmaligem Betätigen des Knopfes bzw. der f1-Taste, legen Sie auch die untere linke Ecke des Ausschnittes fest.

Die Ausschnitt-Koordinaten werden in XA,YA,XB,YB festgehalten. Zusätzlich wird für die anderen Routinen das Flag AF gesetzt. Dieses Flag wird erst wieder gelöscht, wenn Sie den Menuepunkt "Ausschnitt aus" angewählt haben.

Sie sehen also, wie man mit einfachen Mitteln bereits recht ordentliche Ergebnisse erreichen kann. Lassen Sie sich doch einmal etwas einfallen, wie Sie das obige Programm erweitern. Sie brauchen im wesentlichen nur die DATA-Zeilen für einen weiteren Menuepunkt zu ergänzen und die Untermenue-Routinen an das Programm anzuhängen, die Sie dann vielleicht in einer Zeile 495 aufrufen. Viel Spaß!

4. Hardwaregrundlagen

Ihr Commodore 64 besitzt eine ungeheure Vielzahl an Möglichkeiten, Graphiken zu erstellen und zu kontrollieren. Da er jedoch keinerlei BASIC-Befehle zur Verfügung hat, um diese Dinge auszunutzen, muß alles, was die Graphik betrifft, selbst entweder direkt in Maschinensprache oder von BASIC aus programmiert werden, es sei denn, Sie besitzen eine entsprechende Graphikerweiterung, die Ihnen diese Befehle an die Hand gibt. Doch keine noch so gute Erweiterung kann auf alle Fähigkeiten des CBM 64 eingehen. Deshalb bedarf es einer guten Kenntnis der in Ihrem Rechner verwirklichten Graphikorganisation, um sie alle zu nutzen. Außerdem ist es äußerst interessant, festzustellen, wie perfekt die einzelnen Dinge zusammenspielen oder wo es manchmal ärgerliche Haken oder Ösen gibt, durch die man hindurchschlüpfen kann.

Bevor wir uns also mit der Programmierung der Graphik und den vielen Anwendungsmöglichkeiten beschäftigen, die den Nutzen dieser Rechnereigenschaft erst richtig zum Ausdruck bringen (auch oder gerade bei der Verwendung einer entsprechenden Erweiterung, die auf jeden Fall zu empfehlen ist), wollen wir daher eine detailierte Kenntnis über die einzelnen Positionen der Graphik und Ihre Organisation vermitteln. Zugegeben, es ist nicht ganz einfach, aber wir wollen versuchen, es Ihnen so nahe wie möglich zu bringen.

4.1 Die Register des VIC

Zunächst einmal werden Ihnen die 47 Register des VIC, also des Video Interface Chips, dem zentralen Prozessor, der u.a. die gesamte Bildschirmausgabe steuert, kurz und übersichtlich beschrieben. Mit Hilfe dieser Register werden (fast) alle Funktionen bezüglich Graphik, Text usw. gesteuert. Sie besitzen daher fundamentalsten Wert für Ihr Verständnis über Graphik und Bildschirmmanipulationen und sollten schon (bis auf einige Kleinigkeiten) weitgehendst verstanden werden.

Diese Übersicht wird dann in den folgenden Abschnitten und Kapiteln vertieft und näher erläutert. Wenn Sie also nicht sofort alles verstehen, so ist das nur verständlich und praktisch zwingend. Wir werden jedoch stets mit diesen Dingen arbeiten, weswegen Sie im Anhang des vorliegenden Buches noch einmal eine kurze Übersicht über die Registerbelegung jenes integrierten Schaltkreises finden. Grundsätzlich ist diese folgende Beschreibung als Nachschlagewerk

gedacht und auch verfaßt. Nun aber zu den Registern, die mit folgendem Aufbau vorgestellt werden. Zunächst das Register in Dez. und Hex., dann die Kurzbeschreibung, danach die Startbelegung (in Dez. und Dual) sowie eine ausführliche Erklärung:

VIC-Register --- Basisadresse: 53248 ($D000)

---

00 $00 x-Koordinate Sprite 0 00 %0000 0000

Dieses Register beinhaltet die 8 unteren Bits (0-255) der x-Koordinate des ersten Sprites (Sprite 0). Das oberste, 9. Bit (auch MSB = most significant bit genannt) wird dagegen in Register 16 gespeichert. Dies ist notwendig, da die x-Koordinate größer als 255 werden kann.

---

01 $01 y-Koordinate Sprite 0 00 %0000 0000

Wie oben, nur ohne Übertrag.

02-15 $02-0F Koordinaten der übrigen 7 Sprites
(Aufbau wie oben). Sprite 1: Reg. 2/3; Sprite 2: Reg. 4/5 usw.

---

16 $10 MSB der x-Koordinaten 00 %0000 0000

Hier befinden sich die Überläufe (9. Bit) aus den x-Koordinaten Registern. Jedem Bit ist ein Sprite zugeordnet. Bit 0 für Sprite 0 / Bit 1 für Sprite 1 usw.

---

17 $11 Steuerregister 1 155 %1001 1011

Bit 0-2: Bildschirmverschiebung oben/unten
Bit 3: =0: 24 Zeilen / =1: 25 Zeilen
Bit 4: =0: Bildschirm aus / =1: ein

Ist der Bildschirm aus, so wird die CPU nicht mehr vom VIC unterbrochen (was z.B. bei der Generierung von Sprites für bis zu 40 Millisekunden geschehen kann) und Ihr Programm läuft evt. etwas schneller und gleichmäßiger.

Bit 5: =1: Standart Bitmap Mode
Bit 6: =1: Extended Colour Mode
Bit 7: Übertrag aus Register 18 ($12)

---

18 $12 Rasterzeilen-IRQ 55 %0011 0111

Wird dieses Register beschrieben, so geben Sie hier die Bildschirmrasterzeile an, bei deren Aufbau durch den VIC ein IRQ ausgelöst werden kann. Wird es dagegen gelesen, so steht in ihm stets die aktuelle Rasterzeile, die der VIC gerade aufbaut. Der Übertrag dieses Registers steht in Register 17.

---

19 $13 Lightpen-x-koordinate 00 %0000 0000

x-Koordinate (Rasterkoordinate) der Bildschirmposition, die gerade aufgebaut wurde, als ein Signal vom Lightpen kam (Lightpenleitung = 0).

---

20 $14 Lightpen-y-koordinate 00 %0000 0000

Wie Register 19, jedoch y-Koordinate.

---

21 $15 Sprite ein/aus 00 %0000 0000

Hier werden die einzelnen Sprites ein- oder ausgeschaltet. Jedem Bit ist ein Sprite zugeordnet (wie in Register 16).

---

22 $16 Steuerregister 2 08 %0000 1000

Bit 0-2: Bildschirmverschiebung links/rechts
Bit 3: =0: 38 Zeilen / =1: 40 Zeilen pro Zeile
Bit 4: =1: Multicolor Modus
Bit 5-7: unbenutzt

---

23 $17 Spritevergr. y-Richtung 00 %0000 0000

Jedem Sprite ist ein Bit zugeordnet. Bit=1: Sprite wird doppelt so breit. (s. auch Reg. 29)

---

24 $18 VIC-Basisadressen 20 %0001 0100

Hier werden einige der oberen Bits der Startadressen von Video-RAM und Zeichensatzspeicher abgelegt. Durch Änderung ist eine Verschiebung dieser Bereiche möglich (s. auch Reg. 0 der CIA 2). Die Belegung lautet:

Bit 0: unbenutzt
Bit 1-3: Adressbits 11-13 des Zeichensatzes (*2048)
Bit 4-7: Adressbits 10-13 des Video-RAMs (*1024)

Die im CBM 64-Handbuch auf Seite 158 angegebene Belegung dieses Registers ist falsch!

---

25 $19 Interrupt Request (IRR) 15 %0000 1111

Hier kann die Ursache für einen IRQ festgestellt werden:

Bit 0=1: Ursache: Rasterzeilen-IRQ (Reg. 18)
Bit 1=1: Ursache: Sprite-Hintergr. Koll. (Reg. 31)
Bit 2=1: Ursache: Sprite-Sprite Koll. (Reg. 30)
Bit 3=1: Ursache: Lightpen sendet Impuls
Bit 4-6: unbenutzt
Bit 7=1: mindestens eines der ersten 4 Bits ist 1.

Dieses Register muß (soweit es verwendet wird) nach dem Ereignis wieder gelöscht werden. Dies geschieht, indem man den gerade ausgelesenen Wert wieder hineinschreibt.

---

26 $1A Interrupt Mask (IMR) 00 %0000 0000

Hier wird vom Programmierer ausgewählt, durch welches Ereignis ein IRQ ausgelöst werden soll. Die Belegung entspricht der des Registers 25. Ist ein Bit sowohl in diesem wie auch gleichzeitig im Reg. 25 gesetzt, so wird ein IRQ ausgelöst, d.h. alle im Reg. 26 gesetzten Bits ermöglichen die Auslösung eines IRQ durch Reg. 25.

---

27 $1B Priorität 00 %0000 0000

Jedem Sprite ist ein Bit zugeordnet. Bit=1: Hintergrundzeichen hat Priorität vor dem Sprite / Bit=0: Sprite vor Hintergrundzeichen

---

28 $1C Multicolor-Sprites 00 %0000 0000

Jedem Sprite ist ein Bit zugeordnet. Bit=1: Sprite wird im Multicolor Modus gezeichnet.

29 $1D Spritevergr. x-Richtung 00 %0000 0000

Jedem Sprite ist ein Bit zugeordnet. Bit=1: Sprite wird in x-Richtung vergrößert (doppelt so hoch).

30 $1E Sprite-Sprite-Kollision 00 %0000 0000

Jedem Sprite ist ein Bit zugeordnet. Berührt ein Sprite ein anderes, so werden die beiden entsprechenden Bits dieser Sprites gesetzt. Gleichzeitig wird Bit 2 des Registers 25 (IRR) =1. Nach dem Ereignis muß dieses Register gelöscht werden, da sich die Bits nicht selbsttätig zurücksetzen.

31 $1F Spr-Hintergr.-Kollision 00 %0000 0000

Wie Register 30. Hier jedoch wird die Berührung eines Sprites mit einem Hintergrundzeichen (gesetzter Punkt) registriert.

32 $20 Rahmenfarbe 14 %0000 1110

33 $21 Hintergrundfarbe 0 06 %0000 0110

-34-36 $22-24 Hintergrundfarben 1-3 01 02 03

37/38 $25/26 Sprite Multicolor 0/1 04 00

39-46 $27-2E Farbe Sprite 0-7 01 02 03 04 05 06 07

Nach der dezimalen und hexadezimalen Nummernangabe der einzelnen Register, die bei der Ansteuerung stets zu der Basisadresse hinzuaddiert werden muß, folgt, wie Sie sehen, der Registername und die Startbelegung. Unter Startbelegung verstehen wir dabei den Wert, der nach dem Einschalten des Computers in das jeweilige Register als Initialisierungswert eingeschrieben wird. Dieser Wert wird in unserer Übersicht dezimal (z.B. für BASIC-Programmierer) und dual angegeben.

Zusätzlich zu diesen Registern des VIC gibt es natürlich noch einige andere Speicherstellen, die besonders interessant im Zusammenhang mit der Graphikprogrammierung sind. Der Vollständigkeit halber seien sie hier hinten angefügt:

a) Spritedefinitionspointer:

Um dem VIC mitzuteilen, wo in seinem Adressierungsbereich er die 63 Byte lange Definition eines bestimmten Sprites findet, müssen die 8 letzten Bytes des Video-RAM mit einem Pointer belegt werden (im Originalzustand: Speicherstellen 2040 - 2047 bzw. $07F8 - $07FF). Multipliziert man ihn mit 64, so gibt er die Adresse des Definitionsbeginns eines Sprites relativ zu dem Adressbereich des VIC an (einzustellen mit Register 0, Bit 1 und 0 der CIA 2 (s.u.)). Dabei ist jedem der 8 Bytes ein Sprite (von 0-7) zugeordnet und die Pointer können Werte von 0-255 annehmen (16 Kbyte Adressierung).

b) höchstwertige Adressbits des VIC-Adressbereiches

Neben dem VIC-Register 24 gibt es noch eine weitere Speicherstelle, mit welcher z.B. Video-RAM oder Zeichensatz verschoben werden können. Es ist dies das Register 0 der CIA 2 (= Complex Interface Adapter 2) mit der Adresse 56576 ($DD00), speziell Bit 0 und 1. Diese beiden Bits ergeben die obersten zwei Adressbits (Bit 14 und 15) der Basisadresse des VIC. Vorsicht! Die Bits sind LOW-Aktiv, d.h. ist ein Bit=1 so gilt es als 0 und umgekehrt!

c) Joystick/Paddle/Tastatur

Hierfür sind (u.a.) die ersten zwei Register der CIA 1 zuständig:

Register 0 (Adresse: 56320/$DC00):

Normalbetrieb:

Bit 0-7: Reihenauswahl der Tastatur

weitere Aufgaben:

Bit 0-4:	Joystick 0: Bit 0=1: oben
	(Port 1) Bit 1=1: unten
Bit 2=1:	links
Bit 3=1:	rechts
Bit 4=1:	Knopf
Bit 6/7:	Paddle-Set-Auswahl: Bit 6=1: Set A
	(Port 1) Bit 7=1: Set B
	Nur eins der zwei Bits darf = 1 sein!

Für den Joystick-Betrieb muß hier zunächst das Register 0 durch ein POKE 56322,224 (Register 2 - $DC02) auf Eingabe gestellt werden. Rückstellung: POKE 56322,255 (gilt nicht unbedingt für Register 1).

Register 1 (Adresse: 56321/$DC01):

Normalbetrieb:

Bit 0-7:	Spaltenrückmeldung der Tastatur

weitere Aufgaben:

Bit 0-4:	wie Register 0 nur für Port 2
	(Joystick 1 / Paddles)

Soweit in aller Kürze das Wichtigste zur CBM 64-Graphik. Nun lassen Sie sich genauer in die vielen Geheimnisse Ihres Rechners einweihen.

4.2 Die Betriebsarten des VIC

Mit dem Video Interface Chip ist eine große Menge von Einstellungen möglich. Grob unterteilt man diese in drei Kategorien:

- hochauflösende Graphik mit dem Einzelpunktmodus (Standart Bitmap Mode)
- Sprites
- Textmodus (= Zeichen aus einem festen Zeichensatz)

Hinzu kommen noch zwei Modi, die Sie jeweils für diese drei Grunddarstellungen zusätzlich wählen können:

- Normalfarbenmodus
- Multicolormodus

und eine weitere Möglichkeit, die jedoch nur in Zusammenhang mit dem Text- und nicht gemeinsam mit dem Multicolormodus verwendet werden darf, der:

- Extended Color Modus

Die einzelnen Modi sollen im folgenden kurz skizziert werden, um Ihnen einen Überblick zu gewähren. Die nähere Besprechung geschieht dann aber in den späteren Abschnitten (für die Begriffe Farb-RAM, Video-RAM, Graphikspeicher, Zeichensatzspeicher schauen Sie bitte unter Kap. 4.3).

A. Einzelpunktmodus

a) Normaler Einzelpunktmodus

Im normalen Einzelpunktmodus, der durch das Setzen der Bits 5 und 6 des VIC-Registers 17 ausgewählt wird (Register 22, Bit 4=0), besteht ein direkter Zusammenhang zwischen Bildschirm und Graphikspeicher. Ein Bit des Graphikspeichers korrespondiert mit einem Punkt des Bildschirms.

Die Auflösung beträgt 320x200 Punkte, der Graphikspeicher nimmt somit einen Raum von etwa 8 K ein. Die Farbe kommt aus dem etwa 1 K großen Video-RAM, wobei je ein Byte des Video-RAMs die Farbinformation für ein 8x8-Punkte großes Feld des Bildschirms liefert. Dabei gilt für jedes Bit des Graphikspeichers die folgende Farbherkunfts-Zuordnung:

Bit=0: 4 untere Bits des Video-RAMs
Bit=1: 4 obere Bits des Video-RAMs

b) Multicolor-Einzelpunktmodus

In dieser Betriebsart (wählbar durch Setzen der Bits 5 und 6 des VIC-Registers 17 und des 4. Bits des 22. Registers) sind jeweils 2 Bit des Graphikspeichers für einen doppelt breiten Punkt des Bildschirms zuständig. Die Auflösung beträgt daher nur 160x200 Punkte. Dafür sind pro 8x8-Punkte Feld insgesamt 4 verschiedene Farben wählbar. Mit Hilfe der erwähnten 2 Bit pro Punkt wird festgelegt, welche dieser Farben ein Punkt besitzen soll. Dabei gilt folgende Zuordnungen der Farbquellen:

Bits=00: Hintergrund-Farbregister 0
Bits=01: Video-RAM untere 4 Bits
Bits=10: Video-RAM obere 4 Bits
Bits=11: Farb-RAM

B. Sprites

Sprites sind frei definierbare Objekte fester Auflösung, veränderlicher Größe und Farbe, von denen 8 völlig unabhängig voneinander gleichzeitig auf den Bildschirm gebracht werden können. Ihre Priorität untereinander und in Bezug auf die Hintergrundzeichen, sowie Ihre Position auf dem Bildschirm (Bewegungsauflösung: 512x256, also über den Bildschirmrand hinaus) können variiert werden. Die Spritedefinition wird in 63 Bytes untergebracht. Man unterscheidet:

a) Normale Sprites

Diese besitzen eine Auflösung von 24x21 Punkten. Jedes Bit der Spritedefinition repräsentiert einen Punkt der Spritematrix. Für die Farbe gilt dabei:

Bit=0: durchsichtig
Bit=1: Sprite Color Register (Reg. 39-46)

b) Multicolor-Sprite

Bei Multicolor-Sprites bestimmen jeweils 2 Bit der Definition einen doppelt breiten Punkt auf dem Bildschirm. Folglich schrumpft die Punkteauflösung auf 12x21 Punkte. Diese 2 Bit bestimmen die Herkunft der Farbe eines Punktes in folgender Weise:

Bits=00: durchsichtig
Bits=01: Multi Color Register 0
Bits=10: Multi Color Register 1
Bits=11: Sprite Color Register

Es ist möglich, gleichzeitig Sprites beider Darstellungsarten auf dem Bildschirm zu erzeugen.

C. Zeichendarstellung

Bei der Zeichendarstellung wird eine im Zeichensatzspeicher (Zeichengenerator) festgelegte Punkteanordung als festes Zeichen in einer 8x8-Matrix verwendet (für jedes Zeichen sind somit 8 Byte notwendig). In diesem Zeichensatzspeicher sind z.B. alle Buchstaben oder Zahlen festgehalten. Ein Byte des Video-RAM gibt dann die Information, welches der maximal 2x256 Zeichen auf dem Bildschirm erscheinen soll.

a) Normale Zeichendarstellung:

In dieser Betriebsart wird das vollständige Byte aus dem Video-RAM als Zeiger auf ein Bitmuster des Zeichengenerators verwendet. Gleichzeitig sind maximal 256 verschiedene Zeichen auf dem Bildschirm darstellbar. Ein Bit des Bitmusters bestimmt dabei die Farbe eines einzelnen Punktes des Zeichens. Die Farbe stammt aus:

Bit=0: Hintergrundfarbregister 0
Bit=1: untere 4 Bits des Farb-RAM

b) Zeichen im Multicolormodus:

In diesem Modus sind beide Möglichkeiten der Darstellung vorhanden: normale und Multicolor-Darstellung. Ist das 3. Bit des Farb-RAMs =0, so wird das Zeichen in der normalen 8x8-Matrix dargestellt. Wie unter a) beschrieben wird die Farbe des Zeichens ganz normal festgelegt mit der Einschränkung, daß zwangsläufig nur 8 Farben für die gesetzten Bits des Zeichenmusters möglich sind (das 3. Bit des Farb-RAMs ist ja gleich 0).

Ist das besagte 3. Bit jedoch gleich 1, so bestimmen jeweils 2 Bit des Zeichenmusters die Farbe eines doppelt breiten Punktes des Zeichens. Die Auflösung beträgt somit nur 4x8 Punkte. Ein Zeichen aber kann nun aus insgesamt 4 Farben bestehen. Die Herkunft dieser Farben wird durch die beiden Bits eines Punktes bestimmt:

Bits=00: Hintergrundfarbregister 0
Bits=01: Hintergrundfarbregister 1
Bits=10: Hintergrundfarbregister 2
Bits=11: übrige drei Bits des Farb-RAM

Die Farbe drei kann also ebenfalls nur eine von 8 Farben sein. Die restlichen drei Farben sind für alle Zeichen gleich.

c) Zeichen im Extended Color Modus

In diesem Modus kann jedes Zeichen des Bildschirms eine von 4 Hintergrundfarben erhalten. Die Zeichendarstellung an sich entspricht dem Normalmodus (8x8-Matrix). Für die gesetzten Bits des Zeichenmusters (Bit=0) stammt die Farbe ebenso wie im Normalmodus aus dem Farb-RAM. Die gelöschten Bits dagegen, die die Hintergrundfarbe bestimmen, stammen aus verschiedenen Quellen. Die zwei höchsten Bits (Bit 6/7) des Video-RAMs, also des Speichers, der die Zeichen enthält, legen dabei diese Quelle fest:

Bits=00: Hintergrundfarbregister 0
Bits=01: Hintergrundfarbregister 1
Bits=10: Hintergrundfarbregister 2
Bits=11: Hintergrundfarbregister 3

Da aber von den 8 Bits des Video-RAMs nur noch 6 als Zeiger auf das Zeichenmuster übrig bleiben, können nur noch 64 verschiedene Zeichen gleichzeitig auf den Bildschirm gebracht werden!

Soweit der Überblick über die verschiedenen Betriebsarten des Videocontrollers, die in den Kap. 4.4 bis 4.6 näher erläutert werden. Als nächstes folgt eine weitere Spezialität Ihres Commodore 64, die ihn sehr flexibel macht.

4.3 Die Speicherverwaltung des CBM 64

Dieses Kapitel stellt einige Anforderungen an die Kenntnis um Speicheraufbau, Speicheradressierung und ähnliche Dinge und sollte daher von blutigen Laien übersprungen werden. Auch die Leser, die noch nichts über die Graphikmöglichkeiten dieses Gerätes oder gar überhaupt noch nichts über Graphik wissen, sollten sich nach einer kurzen Lektüre des Paragraphen 4.3.2.1 sogleich an das Kapitel 4.4 machen. Um sich allerdings effektiv und vollständig mit der Erstellung von Graphiken zu beschäftigen, ist eine Lektüre der folgenden Seiten unumgänglich, zumal noch einige Aussagen über die allgemeine Speicherhandhabung Ihres Gerätes gemacht werden.

In diesem Zusammenhang sollte darauf hingewiesen werden, daß die meisten und wertvollsten Funktionen nur in Maschinensprache (Assembler) erreichbar sind. Aus diesem Grunde lohnt es in jeden Fall, sich vielleicht einmal mit diesem Themenkomplex zu beschäftigen. Oft besteht eine grundlose Hemmschwelle gegenüber dieser Sprache. Dabei ist Sie für Leute, die bereits BASIC programmieren, relativ leicht zu erlernen, besonders, wenn man ein gutes Buch zur Hand hat. Da DATA BECKER ein wirklich exzellentes Werk über Maschinensprache speziell für 64er-Anwender herausgebracht hat, ist dieser Mangel behoben. Auch das nötige Rüstzeug (Maschinensprachemonitor und Assembler) sind vorhanden. Sie werden sehen, bald programmieren Sie lieber in Assembler, als in dem klobigen und langsamen BASIC.

Da sämtliche Bildschirminhalte gespeichert werden müssen, damit der VIC ständig weiß, was er auf den Bildschirm bringen soll, bedarf es einer Menge an Speicherplatz für Text oder Graphik. Für den Graphikspeicher beispielsweise werden sage und schreibe 8 K, für die dazugehörige Farbe noch einmal 1 K. Dieser Speicherplatz steht dann für andere Zwecke nicht mehr zur Verfügung. Aus diesem Grunde wurden einige Möglichkeiten geschaffen, die Lage dieser Bereiche selbst zu wählen, um optimales Arbeiten zu ermöglichen. Gleichzeitig stehen Bereiche zur Verfügung, die sonst nicht oder nur sehr schwer (z.B. von BASIC aus) genutzt werden.

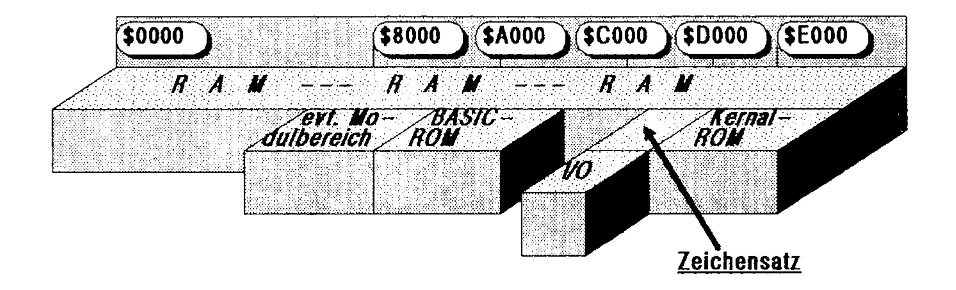
Um die folgenden Ausführungen zu verstehen, muß zunächst einiges über den Speicheraufbau Ihres Rechners gesagt werden:

Der Commodore 64 besitzt einen 6510 als Hauptprozessor, auch CPU (Central Processing Unit) genannt, der für sämtliche Rechenvorgänge, also Programme benötigt wird. Dieser 6510 hat den gleichen Befehlssatz wie sein Vorgänger 6502, ist also softwarekompatibel zu ihm. Doch gibt es beim 6510 einige weitere Leitungen, die unter anderem die Speicherverwaltung unterstützen. Er besitzt damit zwei eigene Register mit den Adressen 0 und 1. Für uns ist lediglich das Register 1 von Bedeutung. Eine weitere Eigenart dieser CPU, die sie diesmal mit dem 6502 gemeinsam hat, ist der sogenannte Adressierungsbereich. Darunter versteht man die Größe des Speichers, den der Rechner ansteuern kann. Dieser beträgt bei Ihrem Rechner 64 K, da für die Adressierung, also die Ansteuerung von Speicherstellen, insgesamt 16 Bit zur Verfügung stehen. Damit können also die Adressen 0-65535 ($0000-$FFFF) angesteuert werden.

Nun wissen Sie aber, daß Ihr CBM 64 allein schon 64 K RAM besitzt (unter RAM = Random Access Memory versteht man Speicher, der gelesen und beschrieben werden kann. Sein Inhalt geht beim Ausschalten des Gerätes verloren. Er dient also als Arbeitsspeicher. Im Gegensatz hierzu kann der sogenannte ROM = Read Only Memory nur gelesen werden. Sein Inhalt wird nicht verändert, auch wenn der Computer ausgeschaltet wird. Seine Aufgabe ist die Beherbergung des Betriebssystems, des BASIC-Interpreters, des Zeichensatzes usw.).

Zu diesen genannten 64 K kommen weiterhin aber noch die verschiedenen ROM-Bereiche und Register der einzelnen Peripheriebausteine (VIC, CIA, Synthesizer etc.), die insgesamt noch einmal einen Platz von 24 K benötigen. Wohin aber mit soviel Speicher, der normal gar nicht angesteuert werden kann?

Die Lösung ist die Speicherüberlappung. Diese wird so realisiert, daß mehrere Speicherbereiche (z.B. ROM und RAM) dieselben Adressen besitzen, also eigentlich an derselben Stelle im Speicher stehen. Der Aufbau sähe dann so aus:



Im gerade eingeschalteten Zustand sind alle diejenigen Bereiche lesbar, die in diesem Schema von unten sichtbar sind. Also:

```
$0000-$9FFF: .i.RAM
$A000-$BFFF: .i.ROM
$C000-$CFFF: RAM
$D000-$DFFF: I/O
$E000-$FFFF: ROM
```

In diese Tabelle schiebt sich bei dem Betrieb eines Moduls noch der Bereich \$8000-\$9FFF als Modul-ROM ein.

Dem Computer muß nun jedoch mitgeteilt werden, welchen Bereich er ansteuern soll. Dies hängt dabei von einigen Faktoren ab. Die für uns wichtigsten sind:

- Lese oder Schreibzugriff des 6510
- Zugriff des VIC oder des 6510
- Ansteuerung des Bereiches v. \$D000-\$DFFF oder nicht
- Inhalt des Registers 1, Bits 0-2 des 6510

Grundsätzlich müssen wir hier zwischen VIC und 6510 unterscheiden, da sie praktisch zur gleichen Zeit unterschiedliche Bereiche ansteuern. Um die Ansteuerung durch den VIC müssen wir uns insofern kümmern, als wir ja bei Graphik o.ä. festlegen müssen, in welche Bereiche wir die einzelnen Speicher (Video-RAM, ...) legen wollen, wir müssen also feststellen, ob der VIC diese überhaupt erreicht.

Die Ansteuerung des 6510 betrifft uns unmittelbar und auch, wenn wir überhaupt nichts mit Graphik bzw. Bildschirmsteuerung zu tun haben. Die Möglichkeiten von BASIC aus sind zwar einigermaßen beschränkt, da viele Bereiche fest in Gebrauch sind, von Maschinensprache (Assembler) aber steht uns alles frei zur Verfügung. Um einen Überblick zu erhalten, beginnen wir mit:

4.3.1 Die Speicherzugriffe des 6510

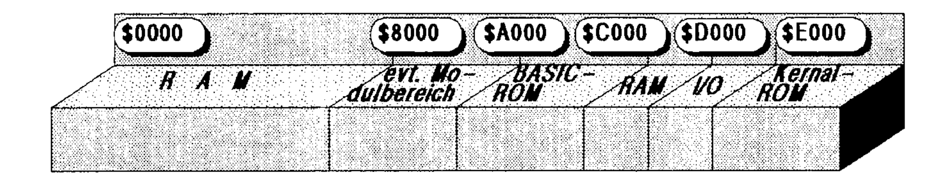
4.3.1.1 Lesen eines Bytes

Soll ein Byte einer bestimmten Adresse (z.B. durch PEEK) gelesen werden (darunter versteht man auch den Ablauf eines Maschinenprogrammes), so hängt die Auswahl, welche der sich überlappenden Speicherbereiche angesprochen werden, - sofern für uns beeinflußbar - nur von dem Inhalt des sogenannten Datenregisters der 6510 CPU (Register 1) ab. Dort haben die ersten 3 Bits (0-2) die Funktion, auf bestimmte Bereiche umzuschalten. Die drei Bits sind nach dem Einschalten des Gerätes gesetzt, was zu der im obigen Schema verdeutlichten Speicherkonfiguration führt, und haben im Einzelnen die folgenden Funktionen:

A. Bit 0/1 - LORAM/HIRAM:

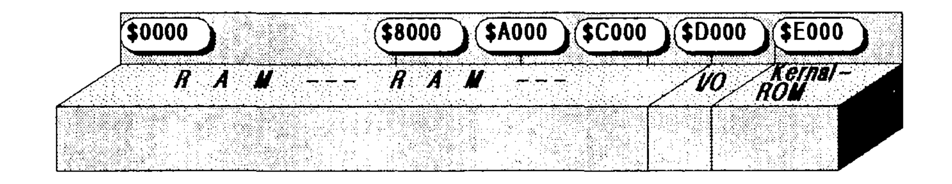
a) Bits 0/1=11:

Sind diese Bits beide 1, so wird bei einem Lesezugriff auf die Adressen $A000-BFFF das dort befindliche BASIC-RAM, bei einem Zugriff auf die Adressen $E000-$FFFF das dortige Kernal-ROM angesteuert. Dies ist der Normalzustand. Ist ein Modul im $8000er Bereich installiert, so ist auch dieses eingeschaltet. Die Speicherkonfiguration sähe dann also so aus:



b) Bits 0/1=10:

Ist lediglich das Bit 1 gesetzt, so ist der Modul- und der BASIC-ROM-Bereich ausgeschaltet und statt dessen wird nun aus dem darunter befindlichen RAM gelesen. Wir haben also folgende Belegung:



Diese Konfiguration kann (auch von BASIC aus) genutzt werden, indem z.B. der BASIC-Interpreter in das unten liegende RAM kopiert wird und, nachdem einige Veränderungen vorgenommen wurden, von nun an in diesem RAM als modifiziertes BASIC liegt. Ein BASIC-Programm dazu könnte z.B. so aussehen:

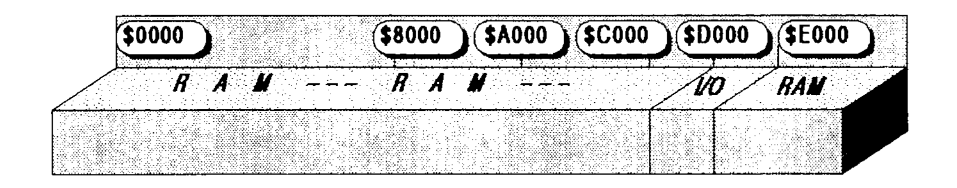
```
10 FOR AD=10*4096 TO 12*4096-1
20 POKE AD, PEEK(X) : REM BASIC-ROM INS RAM KOPIEREN
30 NEXT AD
40 REM HIER NUN VERAENDERUNGEN EINPOKEN ...
50 POKE 1, PEEK(1) AND 254 : REM RAM EINSCHALTEN
```

Zu beachten ist hierbei, daß bei einem Modulbetrieb der Bereich von \$8000-\$9FFF ebenfalls kopiert werden muß, um einen Absturz zu verhindern, da er gleichfalls ausgeschaltet wird.

Gehen Sie mit diesem Register 1 der CPU besonders vorsichtig um (vor allem bei den folgenden Belegungen), da hier jede Änderung tiefgreifende Einschnitte in die Speicherorganisation Ihres Rechners mit sich führt und bei einer Fehlbelegung oft nur der Ausschaltknopf die einzige Möglichkeit ist, Ihren Rechner wieder "zum Leben zu erwecken" (keine Angst, Ihrem Rechner passiert dabei natürlich nichts).

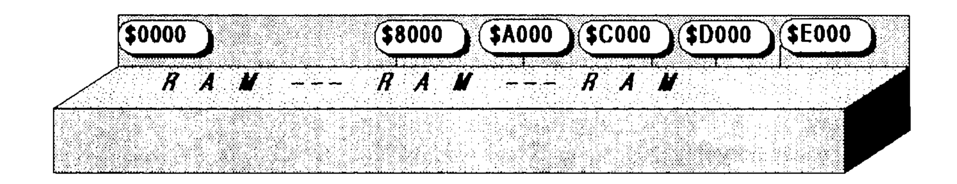
c) Bits 0/1=01:

Ist nun nur das Bit 0 gesetzt, so haben Sie sämtlichen ROM-Speicher ausgeschaltet. Der .i.I/O;-Bereich von $D000-$DFFF bleibt davon jedoch unberührt, kann also weiterhin ohne Änderungen angesteuert werden. Wir erhalten damit 60 K ansteuerbares RAM:



d) Bits 0/1=00:

Sind beide Bits gelöscht, so ist sämtlicher RAM eingeschaltet. Kein ROM und auch keine I/O-Funktionen können mehr ausgelesen oder verändert werden (obwohl der VIC und die anderen Bausteine weiterhin alle ihre Register lesen können und somit das Bild erhalten bleibt!). Die Konfiguration ist denkbar einfach:



Dies ist die einzige Möglichkeit, die unter dem I/O-Bereich und dem Zeichensatz liegenden 4 K RAM mit zu nutzen.

B. Bit 2 - CHAREN:

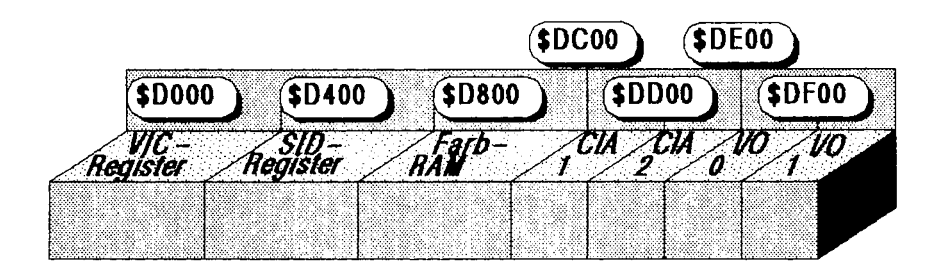
Dieses Bit dient dazu, den verschiebbaren Zeichengenerator (s.u.) auch für den 6510, also für den Programmierer lesbar zu machen, um ihn z.B. zu kopieren und dann zu verändern (s.u.).

a) Bit 2=1:

Dieses Bit bezieht sich lediglich auf den Bereich der $D-Seiten ($D000-$DFFF) des Commodorespeichers. Die anderen Adressen werden nicht beeinflußt. Im Normalzustand liegt es gesetzt vor. Damit kann der 4 K große Zeichensatz, der ebenfalls genau den Bereich von $D000-$DFFF einnimmt, nur vom VIC gelesen werden. Statt dessen stehen die I/O-Adressen, das sind die Register des VIC, des SID (Sound Interface Device) und der CIAs, sowie der Farb-RAM zur freien Verfügung. Wir finden also folgende Einrichtungen in besagtem Raume vor:

b) Bit 2=0

Ist das Bit gelöscht, so kann auch der Programmierer bzw. die 6510 CPU den Inhalt des Zeichensatzspeichers lesen. Dabei belegt er alle 4 K des I/O-Bereiches. Versuchen Sie das jedoch in BASIC, so wird sich Ihr Rechner abrupt innerhalb der nächsten 1/60 Sekunde von Ihnen verabschieden, da 60 mal pro Sekunde eine sogenannte Interruptroutine, von der wir im Kap. 4.7 mehr hören werden, aufgerufen wird, die mit dem in der CIA 1 befindlichen Timer hantiert, der dann natürlich nicht mehr ansprechbar ist, wenn dieses Bit auf 0 gesetzt ist. In Maschinensprache muß zunächst das Interruptflag gesetzt werden (Befehl: SEI), um den Aufruf dieser Interruptroutine zu unterbinden. Später wird durch CLI der Interrupt wieder ermöglicht. Das Gleiche muß geschehen, wenn durch das Löschen von Bit 1 das Kernal-ROM ausgeschaltet wird, in dem sich die Interruptroutine befindet.



4.3.1.2 Schreiben eines Bytes

Wird ein Schreibzugriff auf den Speicher unternommen (z.B. durch POKE), so ändern sich die Verhältnisse grundsätzlich:

Da es nicht sinnvoll ist, etwas in den ROM-Speicher zu schreiben, wurde es eingerichtet, daß jeder Schreibzugriff unabhängig von der Einstellung im Register 1 der CPU den unter dem ROM liegenden RAM erreicht. Mit einer Ausnahme: Der Bereich von $D000 bis $DFFF. Hier wird normalerweise nur der I/O-Speicherbereich erreicht. Wollen Sie auch hier den RAM durch einen Schreibzugriff ansprechen, so stehen Ihnen zwei Möglichkeiten zur Auswahl:

- Ausschalten aller ROMs
- Einschalten des Zeichensatzspeichers

Ersteres geschieht bekanntlich, indem Sie die ersten beiden Bits (0 und 1) des 1. CPU-Registers =0 setzen, der zweite Zustand wird durch Löschen des Bits 2 dieser Speicherstelle erreicht (s.o.).

Nun werden Sie wohl auch das kleine oben angeführte BASIC-Programm vollständig verstehen. In Zeile 20 wurde dort durch den Befehl POKE AD, PEEK(AD) der Inhalt des BASIC-ROMs gelesen, durch den Schreibbefehl (POKE) aber nicht etwa wieder dorthin zurückgeschrieben, sondern in das darunterliegende RAM! Sie sehen, welche Bedeutung dieser Tatsache zukommt.

4.3.2 Die Speicherzugriffe des VIC

Neben dem Hauptprozessor, der für den Ablauf aller Programme zuständig ist, muß selbstverständlich noch der Videocontroller (VIC) auf den Speicher zugreifen, um Bildschirminformationen wie Farbe oder Punktsetzung zu erhalten. Da hier natürlich Zugriffe z.B. auf das Kernal-ROM oder den BASIC-Interpreter sinnlos sind, wurde hier einiges anders gestaltet, als dies bei der CPU-Ansteuerung der Fall ist. Die Umschaltungen zwischen verschiedenen Speicherbereichen nimmt der VIC dabei selbständig vor. Für uns ergibt sich damit ein mehr oder weniger statisches Bild im Vergleich zu den vielen Möglichkeiten des Programmierers bei der gerade besprochenen CPU. Wir müssen also lediglich wissen, welche Bereiche der VIC für welche Zwecke ansteuert und welche nicht. Gleichzeitig zeigt sich die Speicherverwaltung des VIC von einer anderen Seite äußerst dynamisch. Sie selbst können nämlich unter Einhaltung verschiedener Bedingungen bestimmen, wo welche Speicherfunktionen des Videocontrollers liegen sollen.

Mit anderen Worten, Sie haben die Möglichkeit, Graphikspeicher, Zeichensatz, Video-RAM und Spritespeicher in die Speicherbereiche zu verlegen, die Ihnen angenehm erscheinen.

Doch bevor wir uns damit beschäftigen, wollen wir zunächst ein wenig zu den Speicherfunktionen des VIC sagen, um die Begriffe zu klären, die von nun an unser täglich Brot sein werden.

4.3.2.1 Die Speicherfunktionen des VIC:

Um die verschiedenen Aufgaben zu erfüllen, die dem VIC zugeordnet sind (Text, Graphik, Sprites, Farbe, ...) bedarf es umfangreicher Speicherräume mit den unterschiedlichsten Funktionen:

- Zeichengenerator
- Video-RAM
- Farb-RAM
- Graphikspeicher
- Spritedefinitionen

Im Folgenden sollen die Bedeutung und der Nutzen jeder dieser einzelnen Positionen, die schon in früheren Kapiteln erwähnt worden waren, dargelegt und erläutert werden. Die näheren Funktionen und der genaue Aufbau jedoch werden in den folgenden Kap. 4.4 bis 4.6 abgehandelt. Die hier befindliche Auflistung soll lediglich als Orientierung und Begriffserklärung dienen, um das hernach zu Sagende verständlich zu machen, da wir dort ständig mit diesen Dingen umgehen.

a) Zeichengenerator:

Unter Zeichengenerator (auch Zeichensatz oder Zeichensatzspeicher genannt) verstehen wir denjenigen Speicher, der die Definitionen bzw. das sogenannte Bitmuster für jedes einzelne Zeichen enthält, das wir durch einfachen Tastendruck auf den Bildschirm bringen können. Er umfaßt insgesamt 2x2 K und damit die Information für 512 Zeichen, von denen allerdings jeweils nur ein Teil gleichzeitig auf den Bildschirm gebracht werden kann (s. Kap. 4.2, 4.6, 4.4).

b) Video-RAM:

Der Video-RAM umfaßt etwa 1 K und hat verschiedene Aufgaben. Im normalen Zustand (Einschaltzustand) dient er als Zeichenspeicher (nicht zu verwechseln mit "Zeichensatzspeicher"), in dem der sogenannte Bildschirmcode (ein Code für Zeichen ähnlich dem ASCII-Code, er dient als Zeiger auf den Zeichengenerator) für jedes einzelne Zeichen, das sich zur Zeit auf dem Bildschirm befindet, abgelegt ist.

Im Graphikmodus erhält der Video-RAM die Funktion des Farbspeichers, der die Punkt- und die Hintergrundfarbe bzw. in Multicolor Farben 1 und 2 jedes 8x8-Punktefeldes (in MC: 4x8) des Graphikbildschirms bestimmt (s. Kap. 4.4, 4.6, 4.2, 4.4).

Eine weitere Funktion des Video-RAM ist die Beherbergung der Pointer auf die Spritedefinitionen in den letzten 8 Bytes.

c) Farb-RAM:

Der Farb-RAM umfaßt wie der Video-RAM ca. 1 K und liegt fest in dem Bereich $D800-$DBFF (55296-56319). Er ist lesbar und beschreibbar, jedoch sind jeweils nur die unteren 4 Bits aktiv. Die oberen 4 können nicht verändert werden und sind stets gesetzt. Im normalen Modus dient er als Speicher für die Farbe der Textzeichen auf dem Bildschirm. Im Graphikmodus hat er nur bei Multicolor eine Funktion. Dort stellt er die MC-Farbe 3 jeweils für ein 4x8-Punktefeld des Graphikbildes dar.

d) Graphikspeicher:

Der sogenannte Graphikspeicher, der umfangreichste Bildspeicher überhaupt, beinhaltet, wie der Name sagt, den Inhalt eines Graphikbildes. Dabei muß jeder einzelne Punkt des Bildschirms separat gespeichert werden. Bei einer Auflösung von 320x200 (in hochauflösender Graphik) sind dies 64000 Punkte, für deren Speicherung etwa 8 K benötigt werden.

e) Spritedefinitionen:

Um das Aussehen der verschiedenen Sprites zu speichern, werden vom Benutzer diverse Speicherbereiche reserviert, deren Umfang sich jeweils auf 63 Bytes erstreckt. In diesen 63 Bytes sind alle 21x24 Punkte eines normalen Sprite vermerkt.

4.3.2.2 VIC-Speicheransteuerung:

Die Ansteuerung der verschiedenen Speicherbereiche durch den Videocontroller ist weitaus einfacher und übersichtlicher als die vielen Möglichkeiten, die die CPU bietet. Hier kann man einfach nach dem Grundsatz vorgehen: Wenn im folgenden nichts anderes gesagt wird, so wird stets der RAM (auch in dem Bereich von $D000-$DFFF = 53248-57343) vom Videocontroller angesprochen, selbst wenn für den Programmierer lediglich ROM erreichbar ist (also unabhängig von der Einstellung im Register 1 der CPU). Dies ist äußerst wichtig, da dadurch z.B. der Graphikspeicher platzsparend unter den ROM gelegt werden kann, also keinen BASIC-Speicherplatz verschwendet, wie dies z.B. in der SUPERGRAPHIK 64 verwirklicht wurde. Dieser Grundsatz gilt für:

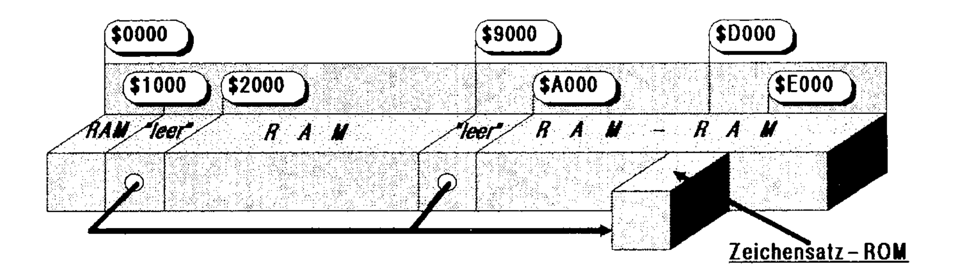
- Video-RAM
- Graphikspeicher
- Spritedefinitionen
- Zeichengenerator

Diese drei Speicher werden also stets aus dem RAM geholt - mit zwei Ausnahmen:

Werden durch eine Einstellung (wie im nächsten Abschnitt beschrieben) die Speicherbereiche von $1000-$1FFF (4096-8191) oder von $9000-$9FFF (36864-40960) angesprochen (etwa, wenn man versucht die Graphikseite nach $8000-$9FFF zu legen, oder in der Einschaltkonfiguration für die Sprites die Blöcke 64-127 (Bereich $1000-$1FFF) zu wählen), so werden nicht etwa die Informationen aus diesen RAM-Bereichen, sondern aus dem Zeichengenerator-ROM geholt, das bei $D000-$DFFF liegt (s.o.).

Diese zunächst merkwürdige Besonderheit erklärt sich aus der Tatsache heraus, daß nach dem Einschalten Ihres Rechners für den VIC nur die unteren 16 K des 64-Speicherbereiches ansteuerbar sind (s.u.). Der feste Zeichensatz, der ja bekanntlich für die Herstellung der einzelnen Textzeichen auf dem Bildschirm notwendig ist, liegt jedoch bei $D000-$DFFF unter den I/O-Adressen. Aus diesem Grunde wurde der Adresse $1000-$1FFF jener Sonderstatus zugesprochen und der Zeichensatz logisch eigentlich nach $1000-$1FFF verlegt. Die anderen Funktionen fallen dem dann leider auch zum Opfer. Der Bereich $9000 resultiert ebenfalls aus dieser Sonderstellung.

Bemerkenswert ist in diesem Zusammenhang, daß bei einer direkten Adressierung (also nicht indirekt über die Adresse $1000) der verschiedenen Funktionen (auch des Zeichensatzes) durch den Programmierer in den Bereich $D000 ff. nicht etwa der dort liegende Zeichensatz-ROM vom VIC angesprochen wird, sondern vielmehr der darunter liegende RAM. Die Speichereinteilung sieht im Schema also folgendermaßen aus:



Dies nur der Vollständigkeit halber. Mehr davon in dem folgenden Abschnitt.

Bei dem Farb-RAM ist die Sache fast noch einfacher: Da er nicht verschiebbar ist, wie die anderen Bereiche (s.u.), wird er stets aus dem Bereich $D800-$DBFF (55296-56319) gewonnen. Hier ist jedoch zu beachten, daß er einen eigenen, vom darunter liegenden RAM verschiedenen Bereich belegt, der für den Programmierer in der Ebene der I/O-Adressen liegt (s.o.). Der Farb-RAM bei $D800 darf also nicht mit dem normalen RAM bei $D800 verwechselt werden.

4.3.2.3 Verschieben der Bildschirmspeicher:

Wohl mit eine der schönsten und praktischsten Dinge in der Speicherverwaltung des Videocontrollers ist die Möglichkeit, die einzelnen Bildschirmspeicher in dem gesamten Speicher Ihres Rechners zu verschieben. Sie können also den Graphikspeicher, den Sie für Ihre Graphiken verwenden, sowohl z.B. nach $2000 (8192) schaffen als auch, wenn Sie die dortige Lage im BASIC-Bereich stört, meinetwegen etwa nach $E000 (41440) unter den ROM verschieben. Vielleicht legen Sie sogar zwei oder mehr Graphikseiten an, von denen Sie dann eine bearbeiten können, während die andere sichtbar für den Beobachter bleibt, wie dies z.B. in der Graphikerweiterung SUPERGRAPHIK 64, die eben schon angesprochen wurde, möglich ist.

Oder Sie verschieben den Zeichensatzspeicher in einen anderen Bereich und können sich so Ihren eigenen ganz persönlichen Zeichensatz erstellen (z.B. mit Umlauten, Sonderzeichen, ... - siehe hierzu auch Kap. 4.6, 4.4). Es eröffnet sich Ihnen eine derartige Fülle von Möglichkeiten, wie Sie sich zur Zeit wahrscheinlich noch gar nicht vorstellen können!

Doch bei jeder Verschiebung halten Sie stets im Auge, welche Speicherbereiche der VIC überhaupt ansteuern kann, was soeben im vorherigen Abschnitt dargelegt wurde. So wird es beispielsweise nie gehen, eine Spritedefinition in den RAM bei $1000-$1FFF zu legen, da dort - wie Sie wissen - der VIC nicht RAM sondern den Zeichensatz-ROM bei $D000-$DFFF ließt (s.o.). Deshalb ist das Verständnis des Paragraphen 4.3.2.2 für die folgenden Ausführungen von unbedingter Notwendigkeit!

a) Allgemeine Verschiebung:

Der VIC oder Videocontroller kann von Haus aus, also intern für sich lediglich 16 K ($0000 - $3FFF oder %0000 0000 0000 0000 bis %0011 1111 1111 1111) adressieren. Unser Adressierungsbereich umfaßt aber 64 K, also 4 mal so viel. Dem VIC fehlen demnach die obersten zwei Adressenbits (Bits 14 und 15). Sie müssen von außen zugeführt werden. Hierfür ist ein Register zuständig, das (selbstverständlich) bereits in Kap. 4.1 erwähnt wurde, es ist das

Register 0, Bits 0/1 der CIA 2 ($DD00=56576)

Diese beiden Bits stellen die gesuchten zwei obersten Adreßbits für die Speicheradressierung des VIC dar (im folgenden Schaubild unterstrichen):

Adreßbits $F E D C B A 9 8 7 6 5 4 3 2 1 0

Man könnte Sie also einfach einsetzen und hätte die vollständige Adresse. Die Sache hat aber einen kleinen Haken. Diese beiden Bits sind LOW-Aktiv, d.h. sind sie gesetzt, so gelten Sie als gelöscht und umgekehrt. Wollen wir die richtige Adresse erhalten, so müssen wir sie erst umdrehen (invertieren). Haben wir dies erledigt, so kennen wir den Bereich, den der VIC nun ansteuern kann, d.h. es verschieben sich automatisch alle Bildspeicherfunktionen, die vom VIC angesteuert werden (außer der Farb-RAM), jeweils in 16 K-Schritten:

- Video-RAM
- Graphikspeicher
- Zeichengenerator
- Spritedefinitionen

Zu Ihrer Unterstützung seien hier die Speicherbereiche tabellarisch festgehalten, die durch eine bestimmte Belegung dieser Bits in die Reichweite des VIC gelangen:

B 0/1	Adr-B	erreichbare Adressen
11	00	$0000-$3FFF (0-16383)
10	01	$4000-$7FFF (16384-32767)
01	10	$8000-$BFFF (32768-49151)
00	11	$C000-$FFFF (49152-65535)

Unter "B 0/1" wird hier die Belegung der zwei Bits aus Register 0 der CIA 2 verstanden. "Adr-B" sind dann die daraus resultierenden Adreßbits 14 und 15 für die VIC-Speicheradressierung. Die originale Belegung ist: B 0/1=11, also der erste Fall in der Tabelle. Nur so kann der Video-RAM von $0400-$07FF (1024-2047) gehen und der Zeichensatz (durch die Sonderstellung der Adresse $1000 (s.o.)) bei $D000 (53248) liegen.

Ein Beispiel: Angenommen, Sie wollen aus irgendeinem Grunde den Video-RAM, der ja die Speicherung aller Bildschirmzeichen vornimmt, nach $C400 (bzw. 50176 = 49152+4*256) verschieben (abgesehen einmal davon, daß ohne weitere Änderungen dann keine Zeichen mehr auf dem Bildschirm verändert werden können). Zu diesem Zweck geben Sie lediglich den Befehl:

POKE 56576, PEEK(56576) AND 253 OR 0

ein, und schon holt sich der VIC seine Informationen nur noch aus dem Bereich zwischen $C000 und $FFFF, was in unserem Falle zu einem wilden Chaos auf dem Bildschirm führt (nur die Farbe kann richtig verändert werden, wegen der Unverschiebbarkeit des Farb-RAMs), das durch den Befehl

POKE 56576, PEEK(56576) AND 253 OR 3

wieder rückgängig gemacht werden kann (was Sie dort blind in die Tastatur eingeben erscheint natürlich erst nach dem <return> auf dem Bildschirm, falls Sie keinen Fehler gemacht haben.).

b) Verschieben des Video-RAM:

Zusätzlich zu dieser allgemeinen Verschiebung kann u.a. separat auch noch der Video-RAM innerhalb dieses 16 K Adressierungsbereiches in kleineren Schritten verschoben werden. Diese Verschiebung ist möglich durch die Veränderung des VIC-Registers 24 ($18), speziell der Bits 4-7 (s. Kap. 4.1). Diese 4 Bits legen hier einen Teil der Adresse zur Ansteuerung des Video-RAMs fest. Es sind dies die Adreßbits 10-13 ($A-$D). Das folgende Schaubild mag das erläutern:

```
Adressbits $F E | D C  B A | 9 8  7 6 5 4  3 2 1 0
---------------------------------------------------
           CIA2 | Reg.  24 |  V I C - i n t e r n
           B0/1 | Bits 4-7 |
```

Der Video-RAM kann also in 1 K-Schritten innerhalb des gesamten 16 K-Adressraumes verschoben werden. Nach dem Einschalten lauten die 4 besagten Bits: %0001. Aus diesem Grunde liegt unser Video-RAM (in Funktion als Textspeicher) bei $400 (1024). Wichtig ist, daß diese Adresse - anders als die unter a) beschriebene Verschiebemöglichkeit - nur und ausschließlich für den Video-RAM gilt. Bei einer Veränderung dieser Bits bleiben die anderen Speicherbereiche in ihrer Lage unverändert.

Ein Beispiel: Wir wollen unseren Video-RAM, also den Textspeicher, der noch bei $400 (1024) liegt, nach $800 (2048) verlegen (auch hier erwarten wir natürlich Nonsense, da hier der BASIC-Speicher beginnt). Dies erreichen wir mit dem Befehl:

```
POKE 53248+24, PEEK(53248+24) AND 15 OR 2*16
```

Um uns wieder in die heimischen Gefilde, sprich: zu unserem alten Video-RAM zu begeben, drücken wir ein:

```
POKE 53248+24, PEEK(53248+24) AND 15 OR 1*16
```

Diese Verschiebemöglichkeit kann z.B. von Nutzen sein, wenn Sie so große BASIC-Programme verwenden, daß Sie den Speicherbereich von $400-$7FF ebenfalls nutzen möchten, oder Sie verwenden zwei Textseiten, die Sie dann ständig umschalten oder ... oder ... oder ...

c) Verschieben des Zeichengenerators:

Neben dem Video-RAM können Sie auch den Zeichengenerator innerhalb des unter a) gewählten 16 K-Raumes verschieben. Auch hier dient uns das 24. Register des VIC als Zwischenspeicher für einige Adreßbits. Diesmal sind es lediglich 3, die die Adreßbits 11-13 darstellen, weswegen wir den insgesamt 2x2 K großen Zeichensatz lediglich in 2 K-Schritten verschieben können:

```
Adressbits  $F E | D C  B | A 9 8   7 6 5 4  3 2 1 0
----------------------------------------------------
            CIA2 | Reg.24 |   V I C - i n t e r n
            B0/1 | Bit1-3 |
```

Interessant ist, daß auch das Betriebssystem Ihres Computers von diesen 3 Bits Gebrauch macht. Wenn Sie durch die Tastenkombination <C=><shift> auf den alternativen Zeichensatz umschalten, so ändert Ihr Rechner das 11. Bit der Zeichensatzadresse durch Änderung des 1. Bits von VIC-Register 24 (s. auch Kap. 4.6).

Wichtig ist dabei das unter Kap. 4.3.2.2 zu der Sonderstellung des Adressenbereiches von $1000-$1FFF (4096-8191) Gesagte. Wollen Sie den Zeichengenerator verschieben, um z.B. einen eigenen zu betreiben, so ist es vielleicht sehr nützlich, wenn Sie auch den Kap. 4.3.1 gelesen haben.

d) *Verschieben des Graphikspeichers:*

Auch die Lage des Graphikspeichers kann gewählt werden. Da er jedoch 8 K groß ist, paßt er nur zweimal in den 16 K-Adressierungsbereich hinein. Aus diesem Grunde können Sie auch nur ein separates Bit für ihn wählen (außer natürlich der allgemeinen Verschiebung). Es ist dies das 3. Bit des VIC-Registers 24, das nun praktisch zwei Aufgaben besitzt:

- Zeichensatzverschiebung
- Graphikspeicherverschiebung

Wie beim Zeichensatz bestimmt es das 13. Bit der Graphikspeicheradresse:

```
Adressbits  $F E | D | C  B A 9 8   7 6 5 4  3 2 1 0
-----------------------------------------------------
            CIA2 | B |     V I C  -  i n t e r n
            B0/1 | 3 |
```

Liegt Ihr VIC-Adreßbereich z.B. bei \$0000-\$3FFF (0-16383), so können Sie wählen, ob der Graphikspeicher bei \$0000 oder \$2000 (8192) beginnen soll (in diesem Fall empfiehlt sich natürlich zweiteres, da ansonsten Nullseite, Stack usw. in Mitleidenschaft gezogen würden). Alles weitere erfahren Sie im nächsten Kapitel 4.4.

4.4 Punktgraphik

Nach so vielen Einzelheiten über Register und Speicherverwaltung sollten wir uns nun einmal speziell dem Aufbau der hochauflösenden bzw. der Multicolor- Graphik, also der sogenannten Punktgraphik (da jeder Punkt einzeln angesteuert werden kann) widmen - ein hochinteressantes Thema und unumgänglich für jeden Graphikprogrammierer. Dieser Abschnitt (und die vier folgenden) sind die hardwaremäßigen Vorbereitungen auf die im Kapitel 4 dargelegten Programmiermöglichkeiten. Versuchen Sie also, auch wenn Sie nicht alles verstehen sollten, sich in diesen Komplex hinein zu denken und wenigstens in etwa den Aufbau der Graphik gegenwärtig zu haben. Dabei wird sich, wie Sie sehen, die recht komplizierte Farbrealisierung Ihres Gerätes in den verschiedenen Graphikarten besonders später in der Anwendung

als ziemlich schwierig herausstellen. Geben Sie jedoch nicht auf. Ein Buch und ganz besonders eines dieser Art sollte sowieso mindestens zweimal gelesen werden. Mit manchen Passagen werden Sie wahrscheinlich täglich arbeiten, wenn Sie sich näher der Graphik widmen wollen. Fangen wir aber gleich einmal an:

4.4.1 Farben:

Ihr Commodore 64 verfügt über die Möglichkeit, 16 Farben sowohl im Graphikbetrieb als auch (wie sicher schon bekannt) für die Textgestaltung zu verwenden. Diese 16 Farben besitzen jeweils einen sogenannten Farbcode, der als Binärzahl in die verschiedenen Register gespeichert wird, die dem Gerät zur Zeichendarstellung verhelfen. Wird z.B. in das 32. Register des Video Interface Chips der Wert 0 gePOKEd, so nimmt der äußere Bildschirmrahmen die Farbe schwarz an. Im Folgenden sind die den einzelnen Farben zugeordneten Codes aufgelistet (im Anhang finden Sie eine vollständige Tabelle, die sich mit dem gleichen Thema beschäftigt):

Code		Farbe	Code		Farbe
Dez	Hex		Dez	Hex	
0	$00	schwarz	8	$08	orange
1	$01	weiß	9	$09	braun
2	$02	rot	10	$0A	hellrot
3	$03	türkis	11	$0B	grau 1
4	$04	violett	12	$0C	grau 2
5	$05	grün	13	$0D	hellgrün
6	$06	blau	14	$0E	hellblau
7	$07	gelb	15	$0F	grau 3

Diese Tabelle wird Sie ständig in allen Bereichen der Graphik begleiten, sei es, Sie arbeiten mit Sprites, Graphik, Text oder was auch immer. Sie sollten Sie also stets im Auge halten. Über den Einsatz der Farben wird Ihnen in den entsprechenden Kapiteln Auskunft gegeben.

4.4.2 Hochauflösende Graphik (HGR)

Ihr Rechner hat die Möglichkeit, von Haus aus zwei verschiedene Graphikarten zu bedienen:

- hochauflösende Graphik (HGR)
- Multicolorgraphik (MC)

Erstere bietet Ihnen ein Graphikfeld von 320 Punkten in x-Richtung (waagerecht) und 200 Punkten in y-Richtung (senkrecht). Man spricht von einer Auflösung von 320x200. Dies verschafft Ihnen ein Reservoir von insgesamt 64.000 Punkten jeweils in gleicher Dichte verteilt auf Ihrem Bildschirmfenster.

Natürlich muß die Graphik genauso wie der Text oder die Farbe gespeichert sein. Schließlich muß der VIC das auf dem Bildschirm entstehende Bild alle ca. 1/20 Sekunden selbst auf Ihrem Fernseher oder Monitor neu erstellen, damit Sie es ständig beobachten können (s. Lightpenkapitel). Dies geschieht für die Punktgraphik mit Hilfe des sogenannten Graphikspeichers. Jeder Punkt ist einzeln ansprechbar und in diesem Graphikspeicher durch ein Bit repräsentiert. Wie Sie wissen (s. Kapitel 2) ist ein Bit eine Informationseinheit und kann die Werte 1 oder 0 annehmen. Jeweils 8 Bit hintereinander bezeichnet man bekanntlich als ein Zeichen (Wort) bzw. als ein Byte. Ein Byte kann also 2 hoch 8 = 256 verschiedene Werte annehmen. Diese kommen durch die verschiedenen Kombinationen von gesetzten (1) und nicht gesetzten (0) Bits zustande.

Ein Byte repräsentiert also 8 Punkte auf Ihrem Bildschirm. Folglich bedarf es 64000/8 = 8000 Bytes (etwa 8 Kilobytes (8 K)), um den gesamten Bildschirminhalt der HGR zu speichern. Wie Sie vielleicht aus dem Kap. 4.3.2.3 wissen, falls Sie sich es durchgelesen haben, können diese 8 K irgendwo im 64 K-Speicher Ihres Rechners plaziert werden. Haben Sie dies getan (s. Kap. 4.2.1.1), so können Sie anfangen. Vorher aber müssen Sie erfahren, wie denn eigentlich der Aufbau des Graphikbildes aus den Speicherinformationen vonstatten geht:

a) Graphikaufbau:

Man könnte sich vorstellen, daß der Graphikspeicher ein direktes Abbild des Bildschirms ist, also Reihe für Reihe nacheinander alle notwendigen Bytes hintereinander folgen. Doch die Angelegenheit ist nicht so einfach, wie sie ausschaut. Der Graphikspeicher besitzt (das ist hardwaremäßig einfacher zu gestalten) den gleichen Aufbau wie der später beschriebene Zeichengenerator. Ein Zeichen besteht, das kann man hier schon einmal vorwegnehmen, aus 8x8-Punkten. Sie besitzen also eine sogenannte 8x8-Punktematrix (Matrix ist der gebräuchliche Begriff für Raster).

Grundlage des Graphikspeichers ist ebenfalls eine 8x8-Matrix, die jeweils durch 8 Bytes des Graphikspeichers dargestellt wird. Da eine solche Matrix also 8 Punkte hoch und breit ist, passen insgesamt 320/8 = 40 solcher Päckchen in eine Zeile und 200/8 = 25 in eine Spalte (genauso ist der Aufbau im Textmodus). Ein Byte liefert nun die Information für eine 8-Punkte breite Reihe eines solchen Päckchens. 8 Bytes untereinander (im Speicher natürlich hintereinander) stellen somit diesen Block dar. Dabei bilden die einzelnen korrespondierenden Bits jedes dieser 8 zusammengehörenden Bytes - also Bits der gleichen Nummer oder Wertigkeit - untereinander liegende Punkte. Dies kann anhand eines kleinen Schaubildes verdeutlicht werden:

```
                 Spalte 0          Spalte 1       ....
----------------------------------------------------------
       Bit:   7 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0   usw.
----------------------------------------------------------
Z Byte 0      . . . . . . . .   . . . . . . . .
e Byte 1      . . . . . . . .   . . . . . . . .
i Byte 2      . . . . . . . .   . . . .
l Byte 3      . . . . . . . .
e Byte 4      . . . . . . . .
  Byte 5      . . . . . . . .
0 Byte 6      . . . . . . . .
  Byte 7      . . . . . . . .
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Z Byte 320    . . . . . . . .   . . . . . . . .
. Byte 321    . . . . . . . .   . . . .
1 Byte 322    . . . . . . . .
  usw.
```

Es wird also Zeile für Zeile aus diesen Päckchen generiert. Noch etwas zu Terminologie: Eine Spalte nennt man die 8 Punkte dicken senkrechten "Balken", von denen jeweils 40 nebeneinander auf den Bildschirm passen. Eine Zeile ist das gleiche, nur waagerecht ausgerichtet. 25 Zeilen passen somit untereinander in das Graphikfenster. Jede Spalte besteht aus 8 senkrechten Reihen, jede Zeile aus 8 waagerechten Reihen. Damit passen 320 senkrechte und 200 waagerechte Reihen in ein Bild. Jede Spalte, Zeile und Reihe ist numeriert (startend bei 0) und kann so genau lokalisiert werden. Diese Vereinbarungen sind im folgenden wichtig, um Verwechslungen zu vermeiden.

Um nun auf einfache Weise einen einzelnen Punkt anzusprechen, gibt man am besten jedem Punkt eine sogenannte Koordinate. Dies sind zwei Werte (x und y), die die Nummer der senkrechten (x-Koordinate) und die der waagerechten Reihe (y-Koordinate) angeben, in denen sich der Punkt befindet. Damit ist jeder Punkt des Bildschirms eindeutig bestimmt. Um nun aus dieser Koordinate die Position des entsprechenden Bytes und Bits im Graphikspeicher zu berechnen, behilft man sich einer entsprechenden Formel, die im Kap. 4.2 unter "Zeichnen eines Punktes" angeführt und erläutert wird (vielleicht versuchen Sie es interessehalber einmal selbst. Das Schema der Zeilenanfangsadressen des Graphikspeichers im Anhang sollte Ihnen dabei eine gute Hilfe sein).

Zum Schluß noch eine Bemerkung: Wie Sie vielleicht schon bemerkt haben, werden nicht alle Bytes der genannten 8 K für den Graphikspeicher verwendet, genauer gesagt nur 8000 ($1F40). Die restlichen 192 ($C0) Bytes können von Ihnen für andere Zwecke genutzt werden. Ähnliches gilt, wie Sie sehen werden, auch für die anderen Speicherbereiche.

b) Farbaufbau:

Zu jedem hochauflösenden Bild, d.h. zu jedem Graphikspeicher, gehört auch ein Farbspeicher. Dieser wird durch den sogenannten Video-RAM dargestellt. Der Video-RAM dient normalerweise dazu, Text oder allgemein Zeichen, die durch Tastendruck auf dem Bildschirm erscheinen, zu speichern, damit der VIC sein Bild erstellen kann. Wenn Sie ihn also nicht verschieben und somit dort lassen, wo normalerweise der Text gespeichert ist, dann erscheinen die verschiedenen Zeichen des ursprünglichen Text-Bildschirms als kleine Farbquadrate auf dem Graphikbildschirm (Über die Verschiebemöglichkeiten von Video-RAM usw. gibt Ihnen Kap. 4.3.2.3 Auskunft). Diesen Effekt können Sie bei dem folgenden Programm beobachten:

```
100 V = 53248 : REM BASISADRESSE VIDEORAM ($D000)
110 REM
120 REM *************
130 REM ** TEIL 1 **
140 REM *************
150 REM
160 REM GRAPHIK EINSCHALTEN:
170 POKE V+17, PEEK(V+17) OR 6*16 : REM BITS 5 UND 6 VIC-REGISTER 17
SETZEN
180 REM GRAPHIKSPEICHER NACH $2000 (8192) VERSCHIEBEN:
190 POKE V+24, PEEK(V+24) OR 8 : REM BIT 3 VIC-REGISTER 24 SETZEN
200 REM WAIT 198,255 : GOTO 220 : REM AUF TASTE WARTEN
210 END
220 REM
230 REM *************
240 REM ** TEIL 2 **
250 REM *************
260 REM
270 REM GRAPHIK AUSSCHALTEN:
280 POKE V+17, PEEK(V+17) AND 9*16+15 : REM BITS 5 UND 6 VIC-REGIS TER
17 LOESCHEN
290 REM ZEICHENGENERATOR RUECKSETZEN:
300 POKE V+24, PEEK(V+24) AND 15*16 + 7 : REM BIT 3 VIC-REGISTER 24
LOESCHEN
310 END
```

Dieses Programm können Sie in zwei Variationen laufen lassen:

1.) so, wie es hier steht
2.) indem Sie das erste REM in Zeile 200 weglassen

Im ersten Fall endet das Programm sofort nach dem Umschalten und Sie können weitere Eingaben machen, die Sie nun allerdings nicht mehr normal sehen, sondern - wie gesagt - jeden Buchstaben als Farbquadrat. Wollen Sie wieder auf normalen Text zurückschalten, so geben Sie ein:

```
RUN 220
```

Im zweiten Fall wartet das Programm auf eine Taste Ihrerseits und schaltet dann die Graphik automatisch wieder aus (s. Kap. 4.2)

In diesem Beispiel wird der Graphik-Farbbezug deutlich. Tatsächlich bestimmt ein Byte des Video-RAM die Farbe für ein 8x8-Punktefeld der HGR. In HGR kann dabei für jedes solches Kästchen sowohl die Hintergrundfarbe, also die Farbe der nicht gesetzten Punkte (oder Bits), und die sogenannte Punktfarbe, d.h. die Farbe der gesetzten Punkte (oder Bits) jeweils aus den 16 verschiedenen Farben gewählt werden. Dabei bestimmen in jedem Byte des Video-RAMs die obersten 4 Bits die Punkt- und die unteren 4 die Hintergrundfarbe dieses Kästchens. Die Farbauflösung ist also weitaus geringer als die normale Graphik erlaubte. Dies war insofern notwendig, als es arg zu viel Speicher verschlingen würde, wenn jedem der 64.000 Punkte eine eigene Farbe zugemessen werden könnte (zumal ein normaler Farbfernseher damit sowieso Probleme hätte). Sie bräuchten dafür 64.000 * 4 = 256.000 Bits, also 32 K RAM. Die Bearbeitungsgeschwindigkeit wäre ebenfalls erheblich herabgesetzt, was einen eigenen Graphikprozessor notwendig machen würde!

Der Video-RAM ist nun etwas einfacher organisiert, als der Graphikspeicher. Hier werden Byte für Byte und Zeile für Zeile nacheinander in den Speicher abgelegt. Der Aufbau sieht dann so aus:

```
              S p a l t e
     ____________________________________
Zeile |   0  1  2  3  4  5  6  7 ... 39
  0   | $00 01 02 03 04 05 06 07 ... 27
  1   | $1E 1F 20 21 22 23 24 25 ... 4F
  2   | $50 51 52 53 ....
 ...  |
 24   |$3C0 3C1 ...
```

Im hochauflösenden Graphikbetrieb ist jedem Byte des Video-RAM also ein eindeutig bestimmtes 8x8-Feld zugeordnet. Hat man die Speicheradresse eines Punktes (abzüglich der Startadresse des Graphikspeichers), so braucht man diese lediglich durch 8 zu teilen und schon besitzt man die Adresse des korrespondierenden Video-RAM-Bytes, zu der man nun nur noch die Startadresse des Video-RAM hinzuaddieren muß.

4.4.3 Multicolorgraphik (MC):

Bevor Sie sich diesem Abschnitt widmen, sollten Sie sich zunächst einmal mit dem letzten Paragraphen (Kap. 4.4.2) beschäftigt haben, da das in jenem Teil des Buches vermittelte Wissen hier zum Teil vorausgesetzt wird.

Wie Sie dort gesehen haben, besitzt Ihr Commodore 64 eine recht hohe Graphikauflösung. Die Farbe kommt dabei jedoch (trotz 16 verschiedener Töne) etwas zu kurz. Um dieses speicherplatzbedingte Manko auszugleichen, haben sich die Konstrukteure entschlossen, einen zweiten Graphikmodus einzuführen, den eine größere Farb-, dafür allerdings eine niedrigere Graphikauflösung auszeichnet: Den Multicolormodus.

Der Multicolormodus ermöglicht es, in einem 8x8-Block statt zwei, insgesamt 4 Farben gleichzeitig zu verwenden. Auch hier findet der 8 K-Graphikspeicher Verwendung. Nun werden allerdings jeweils 2 Bit jedes Bytes für die Bestimmung eines doppelt breiten Bildschirmpunktes benötigt. Die Auflösung beträgt demnach 160 doppelt breite Punkte in x-Richtung (doppelt breit deshalb, da ansonsten das Bildschirmfenster natürlich nur halb so groß wie normal wäre) und 200 Punkte in y-Richtung (Auflösung: 160x200).

Die Farbe stammt nun nicht mehr lediglich aus dem Video-RAM, sondern es werden gleichzeitig noch das Hintergrundfarbregister 0 des VIC und der Farb-RAM hinzugezogen. Wir unterteilen diese Bereiche in 4 sogenannte Farbkanäle, die von 0 bis 3 durchnumeriert sind. Jedes Bitpaar, das ja für einen Punkt zuständig ist und damit Werte von 0-3 (%00-%11) annehmen kann, teilt dem VIC nun mit, aus welchem Kanal er die Farbe des Punktes beziehen soll (In HGR war es bekanntlich so, daß das eine zuständige Bit angab, ob die Farbe aus dem Hintergrundkanal oder dem Punktfarbkanal stammte). Im Graphikspeicher steht also nicht, welche Farbe (Farbcode) ein Punkt haben soll, sondern vielmehr, wo dieser eigentliche Farbcode steht. Die Bitpaar-Kanal-Speicher-Beziehung wird in dem angefügten Schema verdeutlicht:

Bitcode	Kanalnr.	Speicherbereich des Kanals
00	0	Register 33 des VIC
01	1	untere 4 Bits des Video-RAM
10	2	obere 4 Bits des Video-RAM
11	3	Farb-RAM

Unter Bitcode verstehen wir hier die Binärzahl, die die zwei Bits darstellen, die jeweils für einen Punkt zuständig sind. Der Speicherbereich eines Kanals ist der Teil des Speichers, der durch einen Kanal bzw. durch eine Bitcodeeinstellung angesprochen wird.

Ein Beispiel: Im Graphikspeicher wird ein Punkt durch die zwei Bits mit den Belegungen 0 und 1 dargestellt. Der resultierende Bitcode %01 spricht den Kanal 1 an und damit die unteren 4 Bits des zuständigen Bytes des Video-RAM. Dieses zuständige Byte ermittelt man auf genau die gleiche Weise, wie unter Kap. 4.4.2 (HGR) dargestellt.

Neu hierbei ist nun, daß die Farbe 3 bzw. der Kanal 3 aus dem Farb-RAM stammt. Dieser Farb-RAM ist normalerweise für die Farbe des im Video-RAM befindlichen Textes zuständig. Da der Farb-RAM nicht verschiebbar ist, kommt es an dieser Stelle zwangsläufig zu Überschneidungen, wenn gleichzeitig Multicolor und Text verwendet werden. Auch hier läßt sich wieder die Adresse des für einen Punkt zuständigen Bytes des Farb-RAMs auf die unter HGR angegebene Weise bestimmen, da er genauso organisiert ist wie der Video-RAM.

Wie in HGR können diese Farben jeweils für ein 8x8- bzw. (wegen der halben Auflösung) 4x8-Punktefeld durch ein zuständiges Byte des Video- oder Farb-RAMs festgelegt werden. Dabei ergibt sich die Möglichkeit, jedem solchen Kästchen seine eigene Farbkombination zuzuweisen. Lediglich Kanal 0, also die Hintergrundfarbe stammt für die gesamte Graphik aus dem Hintergrundfarbregister 0 des VIC (Reg. 33) und ist damit für das ganze Bild einheitlich.

Wichtig ist bei der Erstellung von Multicolor-Graphiken, auf die Verzerrung in x-Richtung zu achten, die durch die doppelte Punktdicke zustande kommt.

Zum besseren Verständnis sei hier noch einmal ein Schema der Speicherstruktur des Graphikspeichers in Multicolor angeführt, das Ihnen das oben Gesagte noch einmal veranschaulichen soll:

```
           |    Spalte 0     |    Spalte 1     | ....
-----------------------------------------------------
      Bit: | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | usw.
-----------------------------------------------------
Z Byte 0   | <-> <-> <-> <-> | <-> <-> <-> <-> |
e Byte 1   | <-> <-> <-> <-> | <-> <-> <-> <-> |
i Byte 2   | <-> <-> <-> <-> | <-> <->         |
l Byte 3   | <-> <-> <-> <-> |                 |
e Byte 4   | <-> <-> <-> <-> |                 |
  Byte 5   | <-> <-> <-> <-> |                 |
0 Byte 6   | <-> <-> <-> <-> |                 |
  Byte 7   | <-> <-> <-> <-> |                 |
-----------------------------------------------------
Z Byte 320 | <-> <-> <-> <-> | <-> <-> <-> <-> |
. Byte 321 | <-> <-> <-> <-> | <-> <->         |
1 Byte 322 | <-> <-> <-> <-> |                 |
  usw.     |                 |                 |
```

In diesem Schema stellen die "<->" jeweils den doppelt breiten Punkt dar, der auf dem Bildschirm erscheint. An dieser Stelle möchte ich Sie noch einmal auf die unterschiedliche Belegung des Video-RAMs bei HGR und MC hinweisen. Zugegebenermaßen wird die Handhabung der Graphik durch diese schwer durchschaubaren Verhältnisse und Unterschiede nicht gerade vereinfacht. Auch scheint mir die Aufteilung der Farbauflösung nicht gerade gelungen, da sich hierdurch, wie wir noch im 4. Kapitel sehen werden, einige Probleme ergeben. Trotzdem läßt sich doch Einiges mit ihr erreichen. Sie werden sehen, schöne Effekte sind an der Tagesordnung.

4.5 Sprites

Eines der hervorstechendsten Merkmale Ihres Commodore 64 sind natürlich die Sprites. Sprites sind eigenständige kleine Graphiken, die unabhängig voneinander und von dem übrigen Bildschirminhalt in dem Text- oder Graphikfenster bewegt werden können. Insgesamt haben Sie die Möglichkeit, 8 Sprites gleichzeitig auf den Bildschirm zu bringen.

Sprites können bezüglich Ihrer Farbe, Ihrer Größe und der Priorität vor den Hintergrundzeichen und auch gegeneinander variiert werden. Sie können Kollisionen zwischen Sprites untereinander und mit dem Hintergrund feststellen. Zudem besitzen Sie auch hier die Möglichkeit, zwischen den beiden Spritemodi

- normal
- Multicolor

zu wählen, wobei Sie dies bei jedem einzelnen Sprite getrennt bestimmen können. All diese Funktionen können sehr leicht mit Hilfe des VIC (Videocontroller 6567) und seinen Registern realisiert werden. Zunächst aber wollen wir uns einmal mit dem Aufbau der Sprites beschäftigen. Da wir es dabei in besonderem Maße mit der Binärarithmetik und den verschiedenen Registern Ihres Computers zu tun haben werden, sollten Sie sich vorher diese Kapitel (Kap. 4.1 und Anhang) zu Gemüte führen oder, falls Sie diese noch nicht richtig verstanden haben, noch einmal konzentriert lesen.

Während der Erörterung des Spriteaufbaus sollten Sie zwei Dinge stets im Kopf behalten:

Sie können (wie gesagt) gleichzeitig insgesamt 8 verschiedene Sprites auf dem Bildschirm darstellen. Jedem Sprite ist eine spezifische Nummer (0-7) zugeordnet, die Sie durch das gesamte Kapitel begleiten wird.

4.5.1 Aufbau und Farbe normaler Sprites

Jedes normale Sprite besteht aus 504 Punkten, die Sie einzeln setzen oder löschen können. Verwendet wird dabei eine 24x21-Punktematrix, d.h. ein Sprite ist 24 Punkte breit und 21 Punkte hoch. Innerhalb dieses Bereiches können Sie nun die unterschiedlichsten Graphiken oder Figuren erstellen.

Um die Definition, d.h. das Aussehen unserer Sprites zu speichern und dem VIC mitzuteilen, bedarf es insgesamt 504/8 = 63 Bytes, da jeder einzelne Punkt als ein Bit abgelegt wird und ein Byte - wie Sie wissen - 8 Bits umfaßt. Da jedes Sprite eine Breite von 24 Punkten besitzt, passen in eine Reihe genau 24/8 = 3 Bytes hinein. D.h. die ersten drei Bytes bestimmen die 24 Punkte der ersten Reihe. Dementsprechend wird mit der zweiten, dritten, vierten usw. Reihe

verfahren. Jeweils 3 hintereinanderfolgende Bytes legen eine Reihe fest. Die nächste Reihe beginnt dann ebenfalls mit dem nächsten Byte. Wir können diesen Sachverhalt in einer kleinen Skizze darlegen:

```
            |Spalte   0|Spalte   1|Spalte   2
-----------------------------------------------
Reihe/Bit:  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0
-----------------------------------------------
 0 Byte  0  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
 1 Byte  3  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
 2 Byte  6  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
 3 Byte  9  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
 4 Byte 12  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
                          usw.
20 Byte 60  |. . . . . . . . .|. . . . . . . . .|. . . . . . . . .
```

Wollen Sie also ein Sprite eingeben, so geben Sie die zuständigen Bytes Spalte für Spalte und Reihe für Reihe nacheinander ein. Wie dies in BASIC realisiert wird, erfahren Sie in dem großen Kapitel 5.

Sie können den insgesamt 8 verschiedenen Sprites, die Sie gleichzeitig auf den Bildschirm bringen können, jeweils unterschiedliche der zur Verfügung stehenden 16 Farben zuordnen. Dabei erhalten alle gesetzten Punkte die Farbe aus dem für das jeweilige Sprite zuständigen Spritefarbregister (VIC-Register 39-46). Wollen Sie also Sprite 5 weiß zeichnen, so belegen Sie das VIC-Register 39+5 = 44, also die Speicherstelle $D02C (53292) mit dem Wert 1 (für weiß). Alle nicht gesetzten Punkte wirken transparent d.h. durchsichtig und sind daher nicht zu sehen.

4.5.2 Aufbau und Farbe eines Multicolorsprites

Nicht jedes Sprite besitzt diesen 24x21-Punkte-Aufbau. Sie können jeweils zwischen hochauflösenden und sogenannten Multicolor-Sprites wählen. Erstere besitzen die gerade geschilderte Matrix und die dazugehörige Farbgebung. Die Multicolor-Sprites hingegen werden ähnlich der Multicolor-Graphik gebildet. Aus diesem Grunde besitzt ein solches Sprite in x-Richtung die halbe Auflösung. Hier befinden sich also nur 12, jedoch doppelt breite Punkte in einer Zeile. Dafür aber kann ein Gebilde aus insgesamt vier verschiedenen Farben (mit der Hintergrundfarbe) zusammengesetzt sein, während - wie gesagt - ein hochauflösendes (HGR-) Sprite nur zwei Farben beinhaltet. Diese vier

Farben sind vergleichbar mit den Kanälen der MC-Graphik und in den verschiedenen VIC-Registern untergebracht.

Um für jeden der 12x21 = 252 Punkte eines MC-Sprites den Farbkanal zu bestimmen, aus dem die Farbe dieses doppelt breiten Punktes stammen soll, werden jeweils 2 Bits (Bitcode) verwendet, die bekanntlich die Werte 0-3 (%00-%11) annehmen können und damit die Nr. des Kanals festlegen. Der VIC holt sich dann aus diesem Kanal die Farbe des Punktes. Die Zuordnung der Kanäle zu den einzelnen Bitcodes demonstriert die folgende Tabelle:

Kanalnr.	Bitcode	Farbspeicher
0	00	durchsichtig
1	01	Multicolor Reg. 0 (VIC-Reg 37)
2	10	Multicolor Reg. 1 (VIC-Reg 38)
3	11	Sprite Color Reg. (Reg. 39-46)

Wie Sie sehen, kann lediglich die Farbe des Kanals 3 für alle 8 Sprites unterschiedlich sein, da diese für jedes Sprite in einem eigenen Register steht. Die anderen Farben (Farben 1 und 2 neben der Hintergrundfarbe) sind jeweils für alle Sprites gleich, da sie aus identischen Registern gewonnen werden. Entgegen den Angaben des CBM 64 Benutzerhandbuchs können auch in Multicolor sämtliche 16 Farben zur Erstellung Ihrer Figuren verwendet werden.

Um ein Sprite in Multicolor auf dem Bildschirm erscheinen zu lassen, ist es notwendig, dem VIC diese Absicht in einem speziellen Register mitzuteilen. Es ist dies das Register 28 ($1C), in dem jedem Bit des 8 Bit großen Bytes eins der 8 Sprites zugeordnet ist. Ist hier ein Bit gesetzt, so wird von nun an das entsprechende Sprite zu einem Multicolor - Sprite. Die Bitzuordnung dieses Registers ist die folgende:

```
Bit:     b7 b6 b5 b4 b3 b2 b1 b0
Wert:   128 64 32 16  8  4  2  1
Sprite:  s7 s6 s5 s4 s3 s2 s1 s0
```

Wollen Sie z.B. Sprite 4 als Multicolorsprite verwenden, so setzen Sie in dem Register 28 des VIC gleichfalls das Byte 4, d.h. Sie POKEn, sofern Sie von BASIC aus hantieren, wie folgt:

```
POKE 53248+28, 16
```

Wollen Sie mehrere Sprites derart darstellen (z.B. Sprites 1, 5 und 7), so geben Sie beispielsweise ein:

```
POKE 53248+28, 1 OR 32 OR 128  oder:
POKE 53248+28, 1 + 32 + 128
```

(Ausnahmsweise kann der OR-Befehl auch durch eine Addition ersetzt werden) Der Aufbau eines MC-Sprites gleicht ansonsten einem normalen Sprite. Auch jenes wird in 63 Bytes abgelegt. Zur Veranschaulichung ein entsprechendes Diagramm:

```
             | S p a l t e   0 | S p a l t e   1 | S p a l t e   2
-------------------------------------------------------------------
Reihe/Bit:   | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0
 0 Byte  0   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
 1 Byte  3   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
 2 Byte  6   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
 3 Byte  9   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
 4 Byte 12   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
                                      usw.
20 Byte 60   | <-> <-> <-> <-> | <-> <-> <-> <-> | <-> <-> <-> <->
```

Wie Sie sich vielleicht denken können, stellt hier ein "<->" jeweils einen doppelt breiten Punkt dar, der durch ein Bitpaar codiert wird.

4.5.3 Spritedefinition - Farbe

Wollen Sie nun ein Sprite auf den Bildschirm bringen, so haben Sie zunächst einige Dinge zu beachten. Als erstes sollten Sie sich natürlich erst einmal Gedanken über das Aussehen ihres Objektes machen und entsprechend die 63 notwendigen Bytes bereitstellen. Dazu werden Ihnen in Kapitel 5.3 einige Hilfen und Tips gegeben (u.a. ein sehr komfortabler Spriteeditor).

Als Nächstes sollten Sie sich überlegen, wo in Ihrem Speicher Platz für diese 63 Bytes vorhanden ist. Dabei müssen Sie selbstverständlich das unter Kap. 4.3.2 Gesagte mit berücksichtigen. In der Einschaltkonfiguration z.B. gibt es nur relativ wenige Möglichkeiten, Sprites unterzubringen. Hier ist Platz für lediglich 4 verschiedene Spritedefinitionen. Wollen Sie noch mehr unterbringen, so müssen Sie schon entweder einfach den BASIC-Anfang (normal bei $0801 = 2049) verschieben (durch UmPOKEn der Speicheradressen $2B/$2C (43/44)),

wobei Sie allerdings beachten sollten, daß dies vor dem Einspeichern oder Einladen eines BASIC-Programmes geschehen muß, oder Sie verlegen den Video-RAM und haben den alten Video-RAM-Speicher von $400-$7FF zur freien Verfügung, was jedoch bei der gleichzeitigen Textanzeige ein UmPOKEn einer Speicherstelle notwendig macht, die das Highbyte des Video-RAM-Beginns enthält (normal: $04; das Highbyte wird dezimal errechnet durch: INT(Video-RAM-Start / 256). Diese zu verändernde Speicherstelle hat die Adresse $288 = 648. Haben Sie z.B. den Video-RAM nach $0800 (= 2048) verlegt (was nebenbei ebenfalls eine Verlegung des BASIC-Starts notwendig macht) und wollen Sie dort trotzdem Text darstellen, so geben Sie ein:

```
POKE 648, INT(2048/256)
```

Nach diesem UmPOKEn muß dann unbedingt ein <shift><clr/home> bzw. ein PRINT CHR$(147) zum Löschen des Bildschirms folgen.

Doch in den meisten Fällen kommen Sie mit 4 verschiedenen Spritedefinitionen aus, da Sie für zwei gleich aussehende Sprites keine neue Definition abspeichern brauchen, wie Sie gleich sehen werden.

Um dem VIC zu ermöglichen, die von Ihnen abgelegte 63-Byte Definition zu finden und zu lesen, müssen Sie den 16 K-Adressierungsbereich des VIC in 256 Blöcke mit je 64 Bytes unterteilen. Diese Blöcke werden von 0-255 durchnumeriert. Nach dem Einschalten hätten die Blöcke die folgenden Startadressen (bei einer Verschiebung des gesamten VIC-Adreßraumes durch Ändern der Bits 0/1 von Register 0 der CIA 2 (s.o.) muß hier natürlich die neue Basisadresse hinzuaddiert werden):

Block	Startadresse
0	$0000 - 0
1	$0040 - 64
2	$0080 - 128
3	$00C0 - 192
4	$0100 - 256
usw.	
255	$3FC0 - 16320

In jedem solchen Block kann nun eine einzige Spritedefinition untergebracht werden. Dabei hat das letzte Byte des Blockes keine Bedeutung, da ja nur 63 Bytes für ein Sprite benötigt werden. In der normalen Einstellung stehen Ihnen jedoch lediglich die Blöcke:

Block	Adreßbereich
11	$02C0-$02FE (704- 766)
13	$0340-$037E (832- 894)
14	$0380-$03BE (896- 958)
15	$03C0-$03FE (960-1022)

zur freien Verfügung, wobei jedoch angemerkt werden muß, daß die letzten 3 Bereiche sich mit dem Bandpuffer überschneiden, bei dem Arbeiten mit der Datasette also gelöscht werden. Dann beginnt die Möglichkeit des Spritegebrauchs erst wieder bei $2000 (8192), also Block 128 (Vorsicht bei langen BASIC-Programmen und großem Speicherbedarf!), da der Bereich von $1000-$1FFF (4096-8192) bekanntlich dem Sonderstatus unterliegt (s. Kap. 4.3.2.2) und daher nicht benutzbar ist. Bei einer Verschiebung des 16 K-Adreßbereiches gelten natürlich evt. andere Beschränkungen.

Was fangen wir aber mit diesen Dingen an?
Um dem VIC jetzt mitzuteilen, in welchem Block er denn die von Ihnen abgelegte Spritedefinition findet, müssen Sie diese Blocknummer in eines der 8 letzten Bytes des Video-RAMs legen (s. Kap. 4.1). Sie liegen in der Einschaltkonfiguration bei $07F8-$07FF (2040-2047).

Wenn Sie einmal nachrechnen, wieviel Bytes eigentlich für die Speicherung eines Text-Bildschirminhalts benötigt werden, so kommen Sie auf lediglich 40x25 = 1000. Der Video-RAM umfaßt aber genau 1 K, also 1024 Bytes. Die restlichen 24 Bytes werden normalerweise nicht gebraucht und können von Ihnen frei verwendet werden, bis auf die letzten 8 Bytes. Sie werden eben für den gerade genannten Zweck benötigt. Jedem Byte ist dabei ein Sprite in der folgenden Weise zugeordnet (die Adressen gelten für die Position des Video-RAMs nach dem Einschalten und verschieben sich natürlich in dem Falle einer Änderung der Video-RAM-Adresse mit diesem):

Register :	$07F8	07F9	07FA	07FB	07FC	07FD	07FE	07FF
	2040	2041	2042	2043	2044	2045	2046	2047
Spritenr.:	0	1	2	3	4	5	6	7

Ein Beispiel: Angenommen, Sie haben ein Sprite in den Bereich von \$03C0-\$03FE (960-1022), also in Block 15 gelegt. Jetzt wollen Sie, daß sowohl Sprite Nr. 2 wie auch Sprite Nr. 6 so aussieht, wie Sie es in Block 15 definiert haben. In diesem Falle schreiben Sie mittels:

```
POKE 2040+2, 15 : POKE 2040+6, 15
```

den Wert 15 als Zeiger auf Block 15 in die entsprechenden Register ein. Sie sehen, daß auf diese Weise mehrere Sprites das gleiche Aussehen haben können, indem Sie einfach die Zeiger auf den gleichen Block setzen.

Haben Sie nun in Block 14 ein weiteres Sprite definiert und wollen z.B. das Sprite Nr. 2 in seinem Aussehen ändern, so genügt lediglich ein

```
POKE 2040+2, 14
```

um damit schlagartig die Definition zu wechseln (s. Kap. 5.3).

Wir wollen jetzt die beiden Sprites, die wir soeben definiert haben auch auf dem Bildschirm erscheinen lassen. Dazu müssen wir sie jedoch zunächst einmal einschalten. Das VIC-Register 24, "Sprite ein/aus", übernimmt diese Funktion. Jedem der 8 Bits des Registers ist ein Sprite in der gleichen Weise zugeordnet, wie uns dies schon von der Multicolorwahl (Reg. 28) her bekannt ist:

```
Bit:    b7  b6 b5 b4 b3 b2 b1 b0
Wert:   128 64 32 16  8  4  2  1
Sprite: s7  s6 s5 s4 s3 s2 s1 s0
```

In unserem Fall der Sprites 2 und 6 müssen wir also eintippen:

```
POKE 53248+24, 64 OR 4          oder
POKE 53248+24, 64 + 4
```

Doch damit brauchen die Sprites noch nicht sichtbar zu sein, da sie meist erst in den Bildschirm hineinverschoben werden müssen. Um Näheres über die Programmierung der Sprites zu erfahren, sehen Sie bitte unter Kapitel 5 nach.

4.5.4 Weitere Spriteeigenschaften

Doch mit dem einfachen Definieren, der Farbwahl und dem Einschalten haben wir noch längst nicht alles ausgeschöpft, was uns an Gestaltung und Veränderung der Sprites zur Verfügung steht. Die folgenden Zeilen zeigen Ihnen, was die Sprites erst zu den Sprites macht.

4.5.4.1 Positionieren

Die erste Möglichkeit der Variation und wohl auch die wichtigste ist die Wahl der jeweiligen Bildschirmposition jedes einzelnen Sprites. Sie können also bestimmen, wo auf Ihrer Mattscheibe Ihre Figuren zum Stehen kommen sollen. Dies ist besonders wichtig, da dadurch Bewegungen und schöne Effekte erzeugt werden können, wie es im 5. Kapitel dargelegt wird.

Dabei unterteilt man den Bildschirm in sogenannte Koordinaten x und y, wie es uns bereits von der Graphik her bekannt ist. Dabei ist allerdings folgendes zu beachten:

Die Spritekoordinaten stellen stets die Position der unteren linken Ecke eines Sprites dar.

Die Spritebewegung besitzt eine Auflösung von 512x256 Punkten, also weit mehr, als auf dem Bildschirm darstellbar. Das Raster, also der Abstand bzw. die Größe der Punkte, ist dabei identisch mit dem der hochauflösenden Graphik. Es sind also 320 Punkte in x-Richtung und 200 in y-Richtung zu sehen. Damit wird es Ihnen aber möglich, die Sprites jeweils bei Bewegungen aus dem Bildschirm hinausfahren zu lassen; am verdeckenden Rand sehen Sie also einen stets kleiner werdenden Teil Ihres Sprites (s. Kapitel 5).

Der Nullpunkt der Spritekoordinaten liegt weit außerhalb des Text- oder Graphikfensters oben links in der Ecke. Der erste sichtbare Punkt dieses Koordinatenrasters und damit der Nullpunkt der normalen Graphik besitzt schon die Koordinaten x=20 und y=30. Hier erst ist das Sprite also vollständig zu sehen. Um damit von Sprite- auf Graphikkoordinaten umzurechnen, müssen Sie bei ersteren stets 20 vom x- und 30 vom y-Wert abziehen (umgekehrt: Graphik- in Spritekoordinaten umrechnen durch hinzuaddieren dieser Größen). Bei x=320+20=340 und y=200+30=220 ist das Sprite nicht mehr zu sehen.

Um dem VIC nun mitzuteilen, wo er welches der 8 Sprites auf dem Bildschirm unterbringen soll, stehen Ihnen seine ersten 17 Register (von 0 bis 16) zur Verfügung. Das 16. Register nimmt dabei eine Sonderstellung ein und wird etwas weiter unten besprochen. Die 16 zuständigen Speicheradressen sind jeweils paarweise den 8 Sprites zugeordnet. Das erste Element dieser Registerpaare gibt dabei die x-, das zweite die y-Koordinate des entsprechenden Sprites an:

```
Sprite   : s0 s1 s2 s3 s4 s5 s6 s7
x-K. Reg.:  0  2  4  6  8 10 12 14
y-K. Reg.:  1  3  5  7  9 11 13 15
```

Wollen wir also beispielsweise Sprite 6 auf die Koordinaten x=100, y=150 setzen, so brauchen wir lediglich einzutippen:

```
POKE 53248 + 2*6    , 100
POKE 53248 + 2*6 + 1, 150
```

und schon kommt unser vorher definiertes Sprite dort zu stehen. Wie Sie aber vielleicht bereits gemerkt haben, können wir von den oben genannten 512 Punkten der Spritebewegungsauflösung lediglich 256 Punkte erreichen (ein Byte kann maximal 256 verschiedene Werte annehmen). Aus diesem Grunde mußte noch ein weiteres Register eingerichtet werden, das für das oberste 8. Bit (MSB = Most Significant Bit = höchstwertiges Bit) der x-Koordinate jedes Sprites zuständig ist: Register 16. In diesem Byte ist wieder jedem Sprite ein Bit zugeordnet, das die gesuchte Information beinhaltet:

```
Bit:     b7 b6 b5 b4 b3 b2 b1 b0
Wert:   128 64 32 16  8  4  2  1
Sprite: s7 s6 s5 s4 s3 s2 s1 s0
```

Wollen wir also unser Sprite auch über die Koordinate x=255 hinaus auf den Bildschirm bringen, so ist das entsprechende Bit dieses Registers zu setzen. Zu dem Wert im regulären x-Koordinatenregister ist dann 256 hinzu zu zählen.

4.5.4.2 Vergrößerung

So schaut unser Sprite ja schon recht hübsch aus - wir geben uns voll zufrieden - doch Ihr Rechner noch nicht. Er bietet Ihnen einige schöne weitere Kleinigkeiten, die Ihr Herz erfreuen sollen und werden. In dem Registerschatz des Videocontrollers befinden sich nämlich u.a. noch zwei bisher nicht besprochene Adressen. Dies sind die Register 23 und 29. Mif Hilfe dieser beiden Bytes können Sie Ihr Sprite vergrößern. Dabei ist das erste der zwei hier genannten für eine Vergrößerung in y-, also eine Längsdehnung, das zweite für eine Vergrößerung in x-Richtung, also eine Vertikaldehnung, zuständig. Beidesmal ist der Streckungsfaktor gleich 2, d.h. jeder Punkt eines Sprites wird doppelt so hoch bzw. breit. Sie können beide Möglichkeiten getrennt oder gemeinsam (also sowohl Vergrößerung in x- als auch in y-Richtung) anwenden, was jeweils verschiedene Effekte mit sich bringt. Der Aufbau der beiden Register dürfte Ihnen inzwischen bekannt vorkommen und ist in den jeweiligen Diagrammen z.B. unter Positionierung nachzuschauen, da auch hier jedem Sprite ein Bit zugeordnet ist. Ist das entsprechende Bit gelöscht, so wird nicht, ist es gesetzt, so wird vergrößert. Die Verhältnisse können etwa so dargestellt werden:

Vergrößerung	Vergrößerungsfaktor	Punktmatrix
keine	1x1	24x21
x-Richtung	2x1	48x21
y-Richtung	1x2	24x42
x/y-Richtung	2x2	48x42

Unter Punktmatrix ist hierbei natürlich die Matrix gemeint, die auf dem Bildschirm erscheint (sie ist ja auch bei Multicolor-Sprites identisch), also die Anzahl der Punkte des Bildschirms, die von einem Sprite maximal überdeckt werden. Zu bemerken ist weiterhin, daß ein vergrößertes Sprite nicht mehr vollständig am linken oder oberen Rand (oder an beiden) verschwindet, auch wenn die Spritekoordinaten gleich null werden.

4.5.4.3 Priorität

Was passiert nun aber, wenn sich zwei eingeschaltete Sprites oder ein Sprite und z.B. ein Buchstabe überlappen? Überdecken Sie sich und wenn ja, wer verdeckt wen? Dies soll in diesem Abschnitt geklärt werden.

a) Sprite-Sprite-Überlappung:

In diesem Fall ist die Sache denkbar einfach: Den einzelnen Sprites sind bekanntlich Nummern von 0-7 zugeordnet. Überlappen sich jetzt zwei Sprites, so wird dasjenige Sprite "über" dem anderen liegen, es also verdecken, welches die niedrigere Nummer besitzt. D.h. das Sprite z.B. mit der Nummer 0 wird ein Sprite mit der Nummer 5 an den Stellen verdecken, an denen Sie sich überlappen (genau genommen wird das Sprite 0 Sprite 5 nur dort überdecken, wo es nicht transparent (durchsichtig) ist (s. Spritedefinition)).

b) Sprite-Hintergrundzeichen-Überlappung

Bei einer Überlappung eines Hintergrundzeichens mit einem Sprite wird die Sache schon komplizierter, aber auch interessanter. Zunächst aber eine Begriffserklärung: Im folgenden sind unter Hintergrundzeichen stets irgendwelche gesetzten Punkte verstanden, sei es z.B. ein Buchstabe, ein Sonderzeichen oder Graphik.

Wir können, so ist es eingerichtet, hierbei selber wählen, ob ein Sprite von diesen Hintergrundzeichen überdeckt wird, das Sprite sich also praktisch hinter den verschiedenen Objekten des Bildschirms befindet, oder ob es diese selbst verdeckt, also vor Ihnen steht. Für diese Funktion existiert ein weiteres Register des VIC, Register 27. Der Aufbau ist Ihnen wohl inzwischen geläufig, er entspricht dem der Register 16, 21, 23, 28 und 29. Auch hier ist jedem Bit ein Sprite zugeordnet. Ist nun ein Bit gelöscht, was nach dem Einschalten (wie in Kap. 4.1 ersichtlich) der Fall ist, so erscheint das jeweilige Sprite vor den übrigen Zeichen, ist es dagegen gesetzt, so überdecken alle Hintergrundzeichen das betreffende Sprite.

Mit Hilfe dieser wertvollen Eigenschaften ist es möglich, 3-dimensionale Graphik oder Bewegungen darzustellen. In Kapitel 5 wird Ihnen einiges dazu gesagt.

4.5.4.4 Kollisionen

Besonders für Spiele eine unschätzbare Einrichtung: Die Feststellung von Kollisionen bzw. Berührungen zwischen Sprites untereinander und mit Hintergrundzeichen wird Ihnen durch verschiedene andere Register sehr einfach gemacht. Hierfür sind u.a. die Speicherstellen 30 und 31 des VIC zuständig.

In der ersten dieser beiden wird automatisch registriert, wenn sich zwei Sprites im Laufe der Zeit einmal berühren (Überlappung). Dabei ist jedes Bit dieses Registers für eines der acht Sprites zuständig (s.o.). Berühren sich nun zwei Sprites, so werden hier die beiden korrespondierenden Bits gesetzt. Kollidieren also Sprite 6 und 2, so lautet der Inhalt des Registers: %0100 0010. Dieser Inhalt bleibt solange bestehen, bis er (als Zeichen dafür, daß er abgefragt wurde) wieder vom Programmierer z.B. durch

```
POKE 53248+30, 0
```

gelöscht wird. Gleichzeitig mit diesen beiden Bits wird noch ein anderes gesetzt. Es ist dies das Bit 2 des VIC-Registers 25 (IRR), welches uns noch völlig unbekannt ist und auch erst im Kap. 4.7 erläutert wird. Soviel sei gesagt: Falls von Register 26 erlaubt, kann hier also durch eine Kollision ein IRQ (Interrupt Request) ausgelöst werden.

Das zweite Register mit der Nummer 31, das uns in diesem Zusammenhang interessiert, ist für den Vermerk einer Kollision eines Hintergrundzeichens mit einem Sprite zuständig. Findet ein solches Ereignis demnach statt, so wird hier das dem jeweiligen kollidierten Sprite zugeordnete Bit gesetzt (wie Register 30). Berührt z.B. Sprite 2 einen gesetzten Punkt des Bildschirms, so steht hier: %0000 0010. Auch dieses Register muß nach der Abfrage auf dieselbe Art und Weise wieder gelöscht werden wie eben beschrieben. Und auch hier wird in dem Register 25 diesmal das Bit 1 gesetzt, um bei Bedarf einen IRQ auszulösen.

4.6 Text/Zeichensatz

4.6.1 Textseitenorganisation

Damit sich der Rechner alle Ausgaben, die auf dem Bildschirm stehen, merken kann (er muß dieses Bild auf Ihrem Fernseher schließlich alle 1/20 Sekunden selbst erstellen (s. Kap. 4.7)), legt er sämtliche Zeichen, die Sie im normalen Textmodus (z.B. nach dem Einschalten) durch Tastendruck eingeben, in den uns sicher schon bekannten Video-RAM ab. Dieser umfaßt etwa 1 K (in Wahrheit nur 40x25 = 1000 Bytes) und geht im Normalzustand von der Speicherstelle $0400 (1024) bis hin zu $07FF (2047). Über die Verschiebemöglichkeiten gibt Ihnen Kap. 4.3.2 Auskunft. In der hochauflösenden und der Multicolor Graphik wird dieser Speicher für die Beherbergung der Farbe verwendet.

Den einzelnen Zeichen werden jeweils bestimmte Codes zugeordnet und in den Video-RAM abgelegt, wenn dieses Zeichen an einer bestimmten Stelle auf dem Bildschirm erscheinen soll. Die Zuteilung von Codes kennen Sie sicher bereits von der sogenannten ASCII-Codierung. Die Bildschirmcodes jedoch werden nach einem anderen System gebildet. Während Die CBM-ASCII-Tabelle, wie Sie sie im Anhang ihres Bedienungshandbuches finden, manchmal verschiedenen Werten gleiche Zeichen zuordnet und gleichzeitig sogenannte Controlcodes vorhanden sind, die auf dem Bildschirm kein Zeichen erbringen, sind die Bildschirmcodes eindeutig und ohne Lücken verteilt, da Sie neben sämtlichen normalen Zeichen gleichfalls noch die inversen Zeichen unterscheidbar machen müssen - wie anders sollte der Rechner anhand eines Codes wissen, ob er nun ein Zeichen normal oder invers darstellen soll. Wie Sie wissen wird dies von der Tastatur aus durch die Umschaltung mittels zweier Controlcodes (<rvs on> und <rvs off> = CHR$(18) und CHR$(146)) bewerkstelligt. Eine Tabelle der Bildschirmcodes finden Sie im Anhang. Addieren Sie 128 jeweils zu den einzelnen Werten hinzu, so erhalten Sie das gleiche Zeichen in inverser Form.

4.6.1.1 Normaler Text

Neben den Zeichen muß aber für jedes Zeichen gleichfalls die Zeichenfarbe gespeichert werden, da ja bekanntlich rein theoretisch jedes Zeichen eine andere Farbe besitzen kann. Hierfür existiert ein weiterer sogenannter Farb-RAM mit der gleichen Größe wie der Video-RAM, der die notwendigen Informationen enthält. Dieser Farb-RAM liegt bei $D800-$DFFF (55296-56295) und wird ebenfalls in der Multicolor-Graphik als Farbspeicher verwendet. Jedes Byte dieses Bereiches bestimmt die Farbe des dazugehörigen Zeichens des Video-RAMs, dessen Aufbau identisch ist.

Die Hintergrundfarbe des Textbildes wird dagegen durch ein einziges Register des VIC angesprochen. Dieses Register (Register 33) liegt in der Speicherstelle 53248+33 und kann z.B. mit

```
POKE 53248+33,0 : REM HINTERGRUNDFARBE = SCHWARZ
```

verändert werden.

4.6.1.2 Multicolor-Modus

Für den Multicolor-Modus der Zeichendarstellung schauen Sie bitte unter Kap. 4.2 nach, wo dieser entsprechend beschrieben ist.

4.6.1.3 Extended Colour-Modus

Ihr Commodore 64 besitzt neben dem gerade beschriebenen normalen Textmodus mit einer Hintergrundfarbe für alle Zeichen einen weiteren, in dem Sie für jedes Zeichen eine andere Hintergrundfarbe wählen können (jeweils 4 Hintergrundfarbregister, also 4 frei wählbare Hintergrundfarben stehen zur Verfügung). Diese Darstellungsart heißt: Extended Colour-Modus.

Wie gesagt, stehen Ihnen hier für jedes Zeichen eine von 4 Hintergrundfarben zur Verfügung, die Sie durch EinPOKEn der jeweiligen Werte in die folgenden Register verändern können:

```
Hintergrundfarbe 0:  VIC-Register 33
Hintergrundfarbe 1:  VIC-Register 34
Hintergrundfarbe 2:  VIC-Register 35
Hintergrundfarbe 3:  VIC-Register 36
```

Wie Sie sehen, entspricht das Farbregister 0 dem normalen Register zur Festlegung der Hintergrundfarbe. In diese Adressen legen Sie dann natürlich den jeweiligen Farbcode, den Sie dem Anhang entnehmen können (0-15).

Um den Extended Colour-Modus einzuschalten, müssen Sie lediglich etwa durch

```
POKE 53248+17, PEEK(53248+17) OR 4*16
```

das 6. Bit des VIC-Registers 17 ($11) setzen. Durch

```
POKE 53248+17, PEEK(53248+17) AND 256 - 4*16
```

wird es wieder gelöscht.

Wollen Sie nun festlegen, welche dieser Hintergrundfarben die einzelnen Zeichen schließlich besitzen, so müssen Sie folgendes wissen:

Die oberen 2 Bits jedes Bytes aus dem Video-RAM, der eigentlich (wie wir soeben erfahren haben) lediglich die verschiedenen Zeichen des Textfensters speichert, legen dies jeweils für jedes Zeichen fest. Da aber diese Bits normalerweise ebenfalls dazu verwendet werden, um die verschiedenen Zeichen zu codieren, stehen Ihnen im Extended Colour-Modus (ECM) nur 64 Zeichen zur Verfügung.

Alle Graphikzeichen und im Kleinschrift/Großschrift-Modus ebenfalls die Großbuchstaben sowie alle inversen Zeichen fallen dem zum Opfer. Steuern Sie diese trotzdem an, so wird eines der erlaubten 64 Zeichen mit einer anderen Hintergrundfarbe erscheinen. Welche Zeichen davon wie betroffen sind, können Sie am besten der Tabelle der Bildschirmcodes im Anhang entnehmen. Dabei gilt folgende Zuordnung:

MSBs	Bildschirmcodes	Hintergrundfarbe
00	000-063/$00-$3F	HF 0
01	064-127/$40-$7F	HF 1
10	128-191/$80-$BF	HF 2
11	192-255/$C0-$FF	HF 3

Unter MSBs verstehen wir hier die beiden obersten Bits des Video-RAMs, also Bits 6 und 7 (MSB = Most Significant Bit). Vergleichen Sie diese Tabelle mit der angegebenen Bildschirmcodetabelle im Anhang.

Ein Beispiel: Sie wollen ein B mit der Hintergrundfarbe 7 = gelb auf den Bildschirm bringen. Dafür belegen Sie das gewünschte Hintergrundfarbregister (hier 1) mit dem Wert 7 für gelb und POKEn eine 2 für B zuzüglich 64 für die Ansteuerung des Registers 1 an die entsprechende Stelle im Bildschirmspeicher oder geben mittels PRINT-Statement das Zeichen <shift><B>, also CHR$(98) auf dem Bildschirm aus. Im Programm sähe dies dann so aus:

```
10 V = 53248 : REM VIC-REG-BASISADRESSE
20 POKE V+17, PEEK(V+17) OR 4*16 : REM ECM EINSCHALTEN
30 POKE V+34, 7 : REM HINTERGRUNDFARBE 1 = GELB
40 POKE 1024, 2+64 : REM ZEICHEN OBEN LINKS IN DIE ECKE
```

oder:

```
10 V = 53248 : REM VIC-REG-BASISADRESSE
20 POKE V+17, PEEK(V+17) OR 4*16 : REM ECM EINSCHALTEN
30 POKE V+34, 7 : REM HINTERGRUNDFARBE 1 = GELB
40 PRINT CHR$(98) : REM ZEICHEN AN DIE CURSORPOSITION
```

Nach Ablauf dieser Programme befinden Sie sich weiterhin in diesem Modus und können ein wenig herumprobieren. Wie Sie wieder herauskommen, wissen Sie bereits (s.o.).

4.6.2 Zeichensatzorganisation

Neben den ungewöhnlich variationsreichen Graphikmöglichkeiten bietet Ihnen Ihr Commodore 64 noch weitere Kostbarkeiten. Eine dieser Fähigkeiten ist die softwaremäßige Veränderung des Zeichensatzes. Sie ist die Grundlage fast aller Spiele und ist dasjenige Mittel (neben den Sprites), das alle Spiele auf dem CBM 64 so unheimlich schnell und trotzdem graphik- und effektreich werden läßt. Ohne diese Möglichkeit ist eine vernünftige Spielprogrammierung undenkbar geworden. Wo sich andere Computer mit riesigen Graphikspeichern herumquälen, dort schnippst Ihr Commodore 64 einmal mit dem kleinen Finger.

Zunächst einmal eine Definition: Unter Zeichensatz verstehen wir die Gesamtheit aller Zeichen (Buchstaben und Graphikzeichen), die Sie im Textmodus durch Drücken verschiedener Tasten und Tastenkombinationen (siehe <shift> und <C=> (Commmodore-Taste)) auf den Bildschirm bringen können.

Die Form und das Aussehen dieser Zeichen muß dem Computer natürlich bekannt, also irgendwo und irgendwie gespeichert sein. Gleichzeitig sollten Sie auch nach irgendwelchen Kriterien geordnet sein, damit Ihr Rechner weiß, daß er beispielsweise ein B auf den Bildschirm bringen soll, wenn Sie die Taste B drücken. Diese Informationen sind natürlich in allen Computern gespeichert.

Beim CBM 64 ist dieser Speicher so angelegt, daß er softwaremäßig, also vom Programmierer erreichbar ist, d.h. sein Inhalt kann ausgelesen und beispielsweise irgendwo in einen anderen Speicherbereich kopiert (übertragen) werden. Diese Möglichkeit wurde ausführlich in dem Paragraphen 4.3.1 dargelegt. Weiterhin besitzt Ihr Rechner die Fähigkeit, die Speicheradresse, aus der er die Form der einzelnen Zeichen abliest, zu verändern (s. Kap. 4.3.2). Sie haben also die Wahl, Ihrem CBM 64 zu sagen, daß er sich die Zeichengestalt von nun an z.B. aus dem Speicherbereich ab $2000 (= 8192) also aus dem RAM holen soll. Diesen Speicherbereich können wir natürlich selbst verändern.

Haben wir vorher den Zeichensatz aus dem ursprünglichen ROM in diesen Bereich kopiert, so bemerken wir zunächst keine Änderung, da ja alle Information erhalten geblieben ist. Verändern wir jedoch Teile dieses Speicherbereiches, so ändern wir damit gleichzeitig die Form eines bestimmten Zeichens.

Um nun die einzelnen Zeichen nach unseren Wünschen zu gestalten, müssen wir wissen, wie die Form eines Zeichens gespeichert wird. Dies sei im folgenden erklärt:
Ihr Computer besitzt insgesamt 4 Zeichensätze mit je 128 Zeichen, von denen jeweils nur 2 gleichzeitig auf dem Textbildschirm erscheinen. Wir wollen im Folgenden diese vier Zeichensätze kurz benennen:

```
Satz  I/1  -  Normal-Großschrift/Graphikzeichen
Satz  I/2  -  Invers-Großschrift/Graphikzeichen
Satz II/1  -  Invers-Groß-/Kleinschrift
Satz II/2  -  Invers-Groß-/Kleinschrift
```

Bekanntlich können Sie die beiden Zeichensätze I und II durch die gleichzeitige Betätigung der Tasten <C=> und <shift> von Hand aus wechseln. Vom Programm aus dienen hierzu die ASCII-Werte 14 und 142 (Anmerkung: 142 = 128+14), d.h. Sie können mit

```
PRINT CHR$(14);
```

auf Satz II und mit

```
PRINT CHR$(142);
```

auf Satz I umschalten. Mit

```
PRINT CHR$(8);
```

blockieren Sie dabei die Möglichkeit der Umschaltung über die Tastatur, die ja auch während des Programmlaufs möglich ist, und mit

PRINT CHR$(9);

heben Sie diese Blockade wieder auf (s. hierzu auch das CBM 64-Benutzerhandbuch auf den Seiten 135-137).

Die Umschaltung zwischen Sätzen 1 und 2 bewerkstelligen Sie durch die Verwendung von <RVS ON> und <RVS OFF>.

In dem Zeichensatzspeicher müssen natürlich alle diese 4 Zeichensätze getrennt aufgelistet sein. Sie haben also die Möglichkeit 4x128 = 512 Zeichen zu verändern (Wie gesagt, können davon jedoch nur jeweils 256 verschiedene Zeichen gleichzeitig angezeigt werden).

Jedes Zeichen besteht auf dem Bildschirm aus einer Matrix von 8x8 Punkten, wie Sie vielleicht schon wissen. Entsprechend müssen also im Zeichensatzspeicher diese 8x8 = 64 Punkte repräsentiert sein. Das wird erreicht, indem jeder Punkt des Zeichens auf dem Bildschirm - ähnlich wie in HGR - durch ein Bit im Speicher vertreten ist. Somit setzt sich ein Zeichen-Bit-Muster aus 8 Byte zu je 8 Bits zusammen. Jedes Byte repräsentiert eine der 8 Zeilen des Zeichens. Ein gesetztes Bit bedeutet also einen gesetzten Punkt auf dem Bildschirm. Wir können uns die Speicherung eines Zeichens wie folgt vorstellen:

```
   Bit | 7 6 5 4 3 2 1 0
------------------------
Byte 0 | . . . . . . . .
Byte 1 | . . . . . . . .
Byte 2 | . . . . . . . .
Byte 3 | . . . . . . . .
Byte 4 | . . . . . . . .
Byte 5 | . . . . . . . .
Byte 6 | . . . . . . . .
Byte 7 | . . . . . . . .
```

Der Zeichensatzspeicher ist also aus insgesamt 512 hintereinanderliegender Definitionen dieser Art zu je 8 Bytes zusammengesetzt. Er benötigt also einen Speicherbereich von 4 K (= 4096 Bytes), der normalerweise im ROM von $D000 - $DFFF (dezimal: 53248 - 57344) liegt. Dieser Bereich ist jedoch von BASIC aus nicht auszulesen. Zur Demonstration sei an dieser Stelle gezeigt, wie ein normales, großes B im Zeichensatzspeicher definiert wird:

```
   Bit: | 7 6 5 4 3 2 1 0
  ------------------------
Byte 0: | . * * * * . . .
Byte 1: | . * * . . * * .
Byte 2: | . * * . . * * .
Byte 3: | . * * * * . . .
Byte 4: | . * * . . * * .
Byte 5: | . * * . . * * .
Byte 6: | . * * * * . . .
Byte 7: | . . . . . . . .
```

Wir erhalten also folgende 8 Bytes:

```
Byte 0: 0111 1000 = $78 = 120
Byte 1: 0110 0110 = $66 = 102
Byte 2: 0110 0110 = $66 = 102
Byte 3: 0111 1000 = $78 = 120
Byte 4: 0110 0110 = $66 = 102
Byte 5: 0110 0110 = $66 = 102
Byte 6: 0111 1000 = $78 = 120
Byte 7: 0000 0000 = $00 = 000
```

Diese acht Werte stehen nun an der Stelle in dem Zeichensatzspeicher, die für das große, normale B reserviert ist.

Bei Multicolorzeichen stehen jeweils 2 Bit für einen doppelt breiten Punkt des Zeichens, wodurch sich die Auflösung einer Matrix auf 4x8 Punkte pro Zeichen verringert. Da der Vorgang weitestgehendst parallel zu Multicolorgraphik und -sprites funktioniert und da das Wichtigste hierzu bereits unter Kap. 4.2 gesagt wurde, wollen wir lediglich die Punkt-Bit-Beziehung anhand einer kleinen Graphik darstellen:

```
    Bit: | 7 6 5 4 3 2 1 0
  ------------------------
  Byte 0: | <-> <-> <-> <->
  Byte 1: | <-> <-> <-> <->
  Byte 2: | <-> <-> <-> <->
  Byte 3: | <-> <-> <-> <->
  Byte 4: | <-> <-> <-> <->
  Byte 5: | <-> <-> <-> <->
  Byte 6: | <-> <-> <-> <->
  Byte 7: | <-> <-> <-> <->
```

Das nächste Problem ist die Ermittlung der Position einer Zeichendefinition. Ausgangspunkt aller Berechnungen sind die Bildschirmcodes der einzelnen Zeichen, der Codes also, die zur Bestimmung eines Zeichens im Video-RAM stehen (s. Anhang). Der Rest ist relativ einfach: Da jedes Zeichen 8 Byte benötigt, müssen wir nur den Wert des Bildschirmcodes mal 8 nehmen und die Basisadresse des Zeichensatzes, also die Anfangsadresse, bei der unser Zeichensatz beginnt, hinzuaddieren. Bei der Basisadresse ist zu beachten, daß Zeichensatz I und II unterschieden werden müssen. Ein kleines b des Klein-/Großschriftmodus mit dem Bildschimcode 2 liegt 2 K entfernt vom großen B des Großschrift-/Graphikzeichenmodus mit ebenfalls dem Code 2 (nicht zu verwechseln mit dem großen B des Klein-/Großschriftmodus). Im originalen Zeichensatz lauten die Basisadressen:

- Groß/Graphikzeichen: $D000 (53248)
- Klein/Großschrift : $D800 (55296)

Aus soeben Gesagtem ergibt sich die Formel:

adresse = basisadresse + 8 * bildschirmcode

Für das Zeichen B im normalen Zeichenspeicher wäre dieses:

adresse = $D000 + 8 * 2 = $D010 = 53256

Im Kapitel 5 werden Sie ausführlich erfahren, wie Sie diese Änderungen am besten vornehmen (u.a. mit einem sehr komfortablen Zeicheneditor) und wie Sie das Gelernte anwenden können.

4.7 IRQ-Möglichkeiten

Eine der goldenen Eigenschaften Ihres Gerätes ist die mannigfaltig verwendete Interrupt-Einrichtung. Unter Interrupt versteht man die gesteuerte Unterbrechung eines Programms an einer beliebigen Stelle, verursacht durch irgendein Ereignis. Ist eine Unterbrechung ausgelöst worden, so springt der interne Prozessor unterprogrammmäßig (Merken der Rücksprungadressen) an eine bestimmte Stelle im Speicher, die indirekt adressiert wird, und durchläuft dort eine sogenannte Interrupt- oder Unterbrechungsroutine. Nach Beendigung dieser Routine kehrt er genau an dieselbe Stelle des unterbrochenen Programms zurück und fährt ganz normal fort - bis zur nächsten Unterbrechung. Soweit in aller Kürze eine Beschreibung des Interruptvorganges.

Keine Angst, Sie können ruhig weiterlesen, auch wenn Sie von Interrupt noch nie etwas gehört haben und auch der Maschinensprache nicht mächtig sind. Fast alle hier vorgestellten Möglichkeiten, mit Interrupt zu arbeiten, funktionieren gleichfalls ohne diese - nur in Assemblerprogrammen verwendbare - Raffinesse. Falls Sie auf diesem Gebiet also nicht so bewandert sind, sollten Sie sich die Stellen, in denen von Interrupt die Rede ist, trotzdem durchlesen, um einen Überblick zu erhalten. Lassen Sie sich aber nicht abschrecken, da im Anschluß daran stets auch die Möglichkeiten, die von BASIC aus existieren, erläutert werden.

Ein Interrupt ist also eine gewollte und programmtechnisch vorgesehene Unterbrechung (kein Abbruch!). Interrupts sind grundsätzlich ebenfalls in BASIC möglich (z.B. die Supergraphik 64 bietet dieses Hilfsmittel), die hier besprochenen sind jedoch hardwaremäßig eingerichtete (durch Software gesteuerte) Unterbrechungen der CPU (Central Prozessing Unit - Zentraler Mikroprozessor). Nun gibt es für den in Ihrem Gerät verwendeten Mikroprozessor vier verschiedene Interrupts:

- Reset
- NMI (Non Maskable Interrupt)
- BRK (Break)
- IRQ (Interrupt Request)

a) Reset:

Erster, er kann nicht softwaremäßig unterdrückt werden, wird nach dem Einschalten ausgelöst und initialisiert den Computer. Am Ende dieses Vorganges steht die (jedem, der den 64er einmal eingeschaltet hat, bekannte) Kopfschrift auf dem Bildschirm:

```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
```

b) NMI:

Dieser nicht maskierbare, d.h. ebenfalls unbedingte Interrupt wird ausgelöst, wenn Sie auf die <restore>-Taste drücken. Er wird gleichfalls benötigt, um die RS 232-Schnittstelle zu bedienen, falls vorhanden. Sie können die indirekte Sprungadresse des NMI in den Speicherstellen $318/$319 (792/793) erfahren oder ändern.

c) BRK:

Dies ist ein sogenannter Softwareinterrupt, der von einem Assembler-Programm aus betätigt werden kann. Er wird dann ausgelöst, wenn der Prozessor auf den BRK-Code $00 stößt (indirekter Sprung zur der Adresse, die in den Speicherstellen $316/$317 (790/791) steht).

d) IRQ:

Hier ist er endlich! Alle oben genannten Interrupts sollen uns hier nicht weiter interessieren und sind nur der Vollständigkeit halber erwähnt worden. Die eigentlich für uns interessante Unterbrechung ist dieser sogenannte maskierbare Interrupt. Maskierbar heißt, daß per Software gesagt werden kann, ob er ausgelöst werden darf oder nicht. Sie können ihn also beliebig abschalten, wenn es Ihnen gefällt. Hierfür existieren die beiden Assemblerbefehle SEI (Set Interruptflag - verhindert Interrupt) und CLI (Clear Interruptflag - ermöglicht Interrupt). Gleichzeitig können Sie beim Commodore 64 in speziellen Registern auswählen, welche Ursache von einer ganzen Palette an Möglichkeiten einen IRQ auslösen darf und welche nicht. So können beispielsweise die Timer der CIA nach Ihrem Ablauf eine Unterbrechung hervorrufen, was vom BASIC-Betriebssystem genutzt wird, indem es alle 1/60 Sekunde den normalen Programmablauf

unterbricht und in die ROM-IRQ-Routine springt (indirekte Adresse in $314/$315; dezimal: 788/789). Hier wird der Cursor blinken gelassen, die interne Uhr (TI$) gestellt, und die Tastatur (z.B. STOP-Taste) abgefragt. Doch wir können die Auslösung eines IRQ gleichfalls einigen anderen Ereignissen als nur dem Ablauf eines Timers ermöglichen. Die für uns interessanten sind:

- Rasterzeilendurchlauf
- Lightpen
- Sprite-Sprite-Kollision
- Sprite-Hintergrund-Koll.

Wenn wir nun die indirekte Adresse der IRQ-Routine des Betriebssystems auf eine eigene Interruptroutine richten, können wir diese Möglichkeiten ausnutzen. Wie dies programmtechnisch zu lösen ist, wird Ihnen in den entsprechenden Abschnitten des 5. Kapitels dargelegt. Der Vorteil der IRQ-Verwendung für diese Anwendungen ist, daß das Ereignis sofort gemeldet wird, ohne daß bis zur nächsten Abfrage gewartet werden muß (neben dem IRQ gibt es natürlich bei allem die Möglichkeit, auf das Ereignis im Laufe des Programms durch eine einfache Abfrage zu testen). Dies hat besonders große Bedeutung bei dem Rasterzeilen-IRQ, da hier die Vorgänge sehr schnell ablaufen müssen. Nun aber zu den angekündigten Einzelbesprechungen:

4.7.1 Bildschirmrasterzeilen

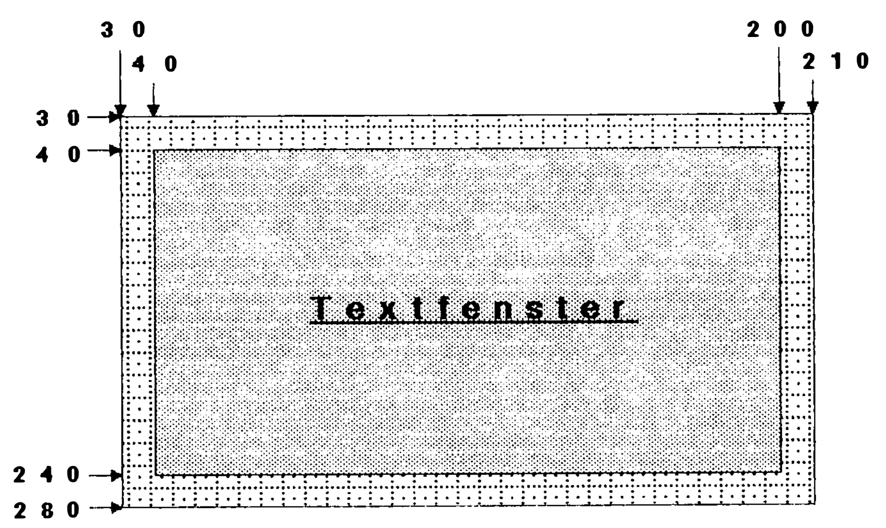
Eine der am wenigsten verstandenen, dafür jedoch äußerst ansprechenden Fähigkeiten Ihres Computers ist der Rasterzeileninterrupt. Sie können mit diesen Dingen eine große Anzahl von Effekten erzielen, die Ihr Herz höher schlagen lassen: Mehrere Hintergrundfarben, mehr als 8 Sprites, gemischte Graphik- und Textanzeige usw. usw. Doch bevor wir uns in Kapitel 4 mit der Realisierung dieser Dinge beschäftigen, wollen wir zunächst einmal die Frage klären, was unter sogenannten Rasterzeilen eigentlich zu verstehen ist.

Dazu müssen wir verstehen, wie auf dem Fernseher oder Monitor ein Bild entsteht. Wie Sie vielleicht wissen, besteht der Bildschirm aus einer Lochrasterplatte in Verbindung mit einer phosphoreszierenden Schicht, die durch den Aufprall der Elektronen eines Elektronenstrahles punktuell zu leuchten beginnt. Dieser Elektronenstrahl geht nun Zeile für Zeile jeden einzelnen Punkt des Lochrasters durch und

läßt ihn - je nach Bedarf - aufleuchten oder nicht. Dies geschieht in einer ungeheuren Geschwindigkeit, so daß der Strahl pro Sekunde 20 mal ein neues Bild erzeugt, also 20 mal über sämtliche Punkte des Punktrasters hinüberfegt. So entsteht für uns der Eindruck eines bewegten Bildes. Dieser Elektronenstrahl wird durch die komplizierten Apparaturen am hinteren Ende einer Bildröhre gesteuert. Doch die Information, ob ein Punkt des Bildschirms nun aufleuchten oder erblassen soll, muß von einer anderen Quelle stammen. Beim normalen Fernseher sind dies die über den Äther gesandten und von der Antenne eingefangenen Signale der Sendestationen. In unserem Fall muß der Computer dieses Signal erzeugen. Er also muß praktisch Reihe für Reihe und Punkt für Punkt durchgehen, und das "an/aus"-Signal senden. Diese Aufgabe übernimmt im Falle des CBM 64 ebenfalls der Videocontroller (VIC).

Normalerweise wird das alles intern geregelt, ohne daß der Programmierer darauf Einfluß nehmen oder überhaupt daran beteiligt sein könnte. Anders beim 64er: Hier besitzt die Software die Möglichkeit festzustellen, welche Rasterzeile der VIC gerade erstellt. Dies kann u.a. durch das Lesen des VIC-Registers 18 erfolgen. Hierzu muß zunächst aber etwas gesagt werden:

Da bei der Rasterzeilenerstellung selbstverständlich neben dem eigendlichen Text- oder Graphikfenster ebenfalls der Rahmen mit übergeben werden muß und die Strahlablenkung etwas über den Bildschirm hinausgeht, stellt sich die Koordinateneinteilung etwas anders dar, als wir sie von der Graphik oder den Sprites her kennen. Zur Veranschaulichung des im folgenden Gesagten vergleichen Sie bitte das unten gezeigte Schaubild, das eine Skizze des Bildschirms mit dem Fenster darstellt.



Der Strahl startet in der obersten Reihe. Diese besitzt nun aber nicht etwa die Nummer 0 oder 1, wie man vermuten könnte, sondern die obere Kante Ihres Bildschirms beginnt bereits bei Reihe Nr. 30 ($1E) (wobei die angegebenen Randwerte von Fernseher (Monitor) zu Fernseher leicht variieren können). Sie endet (untere Bildschirmkante) ca. bei Nr. 280 ($118). Das eigentliche Bildschirmfenster, das normalerweise verwendet werden kann, hat an der oberen Kante den Rasterzeilenwert von etwa 40, während die untere mit der Rasterzeile Nr. 240 übereinstimmt.

Wie Sie sehen, unterteilt der VIC das Textfenster (genau übereinstimmend mit der Punktauflösung) in 200 Zeilen (Reihen). Sie werden im Kapitel "Joystick" sehen, daß dies im Falle der Spaltenauflösung nicht so einfach ist.

Wie gesagt können wir jetzt selber aktiv in das Geschehen eingreifen, bzw. uns über den jeweiligen Stand unterrichten. Was heißt das nun?

Zum einen können wir (wie oben erwähnt) dem Register 18 des Videocontrollers die Nummer der Rasterzeile, die er gerade dem Sichtgerät sendet, entnehmen. Da ein Register jedoch Werte von 0 bis maximal 255 annehmen kann, der VIC aber mindestens bis zu 280 Reihen sendet, wird das fehlende oberste (8.) Bit vom VIC-Register 17 geliefert. Wie Sie aus der in Kap. 4.1 gezeigten Tabelle entnehmen, stellt das 7. Bit dieses Registers 17 das gesuchte oberste Bit der Rasterzeilennummer dar. Somit können wir durch einfaches Lesen die genaue Rasterzeile erfahren. Da jedoch ein Bild pro Sekunde 20 mal erneuert wird, und mindestens 280 Reihen pro Bild an den Bildschirm gesendet werden müssen, wird eine Reihe in ca. 1 sek./(20*280) = 0,00018 sek., also in 0,18 milli- oder 180 mikrosekunden (eine Mikrosekunde ist eine Millionstel Sekunde), das sind 5600 Reihen pro Sekunde, aufgebaut. Wenn man bedenkt, daß eine Reihe weiterhin noch aus einer großen Zahl von Punkten (s. Kap. 4.7.2) besteht, kann man sich in etwa vorstellen, wo hier die Zeitverhältnisse pro Punkt liegen (etwa bei 0,89 Mikrosekunden).

Damit scheint die softwaremäßige Behandlung auch in Assembler (immerhin dauert ein Befehl in Assembler mindestens eine 500.000stel Sekunde (= 2 Mikrosekunden) - bei manchen Befehlen das bis zu 3,5-fache) unmöglich, da praktisch dauernd abgefragt werden müßte, wo sich der Strahl gerade befindet, um eine bestimmte Reihe anzusteuern.

Doch dies ist aufgrund einer äußerst interessanten Einrichtung nicht notwendig. Sie können nämlich den VIC dazu veranlassen, einen IRQ (wie oben beschrieben), also eine Unterbrechung Ihres Programms, auszulösen, wenn er gerade eine bestimmte Reihe des Bildes aufbaut (genau einen bestimmten Punkt anzusprechen, wäre aufgrund der winzigen Punktdurchlaufzeiten (s.o.) sinnlos). Für diesen Zweck schreiben Sie die gewünschte Zeile, bei deren Strahldurchlauf ein Interrupt ausgelöst und damit eine Interruptroutine aufgerufen werden soll, genau in dasselbe Register 18, aus dem wir sonst die aktuelle Position des Strahl ziehen. Auch hier dient Bit 7 des 17. Registers als Highbyte. Wir müssen dem Computer (bzw. dem VIC) nur noch mitteilen, daß er ab sofort diese Unterbrechung ausführen soll, wenn er die bestimmte Reihe erreicht hat. Dies geschieht mit Hilfe der beiden Register 25 und 26 ($19/$1A). Ersteres ist das sogenannte Interrupt Request Register (IRR). Hier wird angegeben, welche Ursache ein durch den VIC ausgelöster IRQ hat. Dabei gilt:

```
Bit 0 = 1: Rasterzeileninterrupt
Bit 1 = 1: Interrupt durch Sprite-Hgrund-Koll.
Bit 2 = 1: Interrupt durch Sprite-Sprite-Koll.
Bit 3 = 1: Interrupt durch Lightpen
Bits 4-6 : unbenutzt
Bit 7 = 1: Interrupt hat eine der 4 Ursachen
```

Sie sehen also, daß hier der Programmierer durch Abfrage des 7. Bits erfährt, ob eins der 4 unteren Bits gleich 1 ist, ein Interrupt also durch eine der 4 Möglichkeiten verursacht wurde (wie gesagt kann er ja auch andere Ursachen haben). Nach jeder Abfrage muß dieses Register wieder zurückgesetzt werden, da ansonsten direkt nach der durchlaufenen Interruptroutine wieder ein IRQ ausgelöst wird usw. - "Absturz"! Dies geschieht, indem man denselben Wert, den man aus diesem Register ausgelesen hat wieder hierhin zurückschreibt.

Trotzdem wüßte der VIC immer noch nicht, wodurch er einen IRQ auslösen sollte, würden wir nicht Gebrauch von dem Register 26 machen. Hier existiert die gleiche Zuordnung der einzelnen Bits wie im gerade beschriebenen Register 25 (außer 7. Bit). Hier bedeutet allerdings ein gesetztes Bit, daß das betreffende Ereignis von Stund an ein IRQ-Auslöser sein kann.

Wollen wir beispielsweise, daß der VIC immer dann unser Programm unterbricht und der Prozessor dann unsere IRQ-Routine anspringt, deren Adresse wir in $314/$315 (= 788/789; s.o.) abgelegt haben, wenn er die Reihe 100 erreicht hat, so schreiben wir zunächst 100 in dieses Register 18 (MSB = 0!), löschen Register 25, indem wir den Inhalt lesen und wieder rückschreiben, und setzen das Bit 0 des Registers 26 gleich 1. Wie das alles programmtechnisch unter einen Hut gebracht wird, sollten Sie nun in dem entsprechenden Paragraphen des 5. Kapitels nachlesen.

4.7.2 Lightpen

Bevor Sie diesen Paragraphen durcharbeiten, sollten Sie wenigstens die Ausführungen in 4.7.1 über die Entstehung des Bildes auf dem Bildschirm gelesen haben, da die folgenden Erklärungen auf diesem Wissen aufbauen.

Ihr Commodore 64 besitzt verschiedene Möglichkeiten, steuernde Geräte als Peripheriebausteine anzuschließen. Zu diesem Zwecke befinden sich an der rechten Seite Ihres Computers, wenn Sie sich das einmal anschauen wollen, zwei sogenannte Controlports. Dies sind Steckbuchsen für den Anschluß von Joysticks, Paddles, Lightpen oder sogar selbstgebaute Steuer- bzw. Meßeinrichtungen (wie z.B. Thermometer, Feuchtigkeitsmesser, Impulsgeber etc.). Die Steckerbelegung wird in Ihrem CBM 64-Benutzerhandbuch auf der Seite 141 beschrieben. Sie brauchen diese nicht unbedingt zu kennen, um beispielsweise einen Joystick an Ihr Gerät anzuschließen und ihn richtig zu gebrauchen. Wichtig ist nur, daß der Eingang für den unten beschriebenen Lightpen identisch ist mit dem des Feuerknopfes eines in Port 1 gesteckten Joysticks. Um etwas vorzugreifen: Sie können also auch mit dem Feuerknopf von Port 1 einen Interrupt auslösen, was sicher hochinteressant nicht nur für Spiele ist. Sie können damit Geräte anschließen, die im Computer einen IRQ auslösen können, eine Möglichkeit, die wertvolle Konsequenzen hat!

Eine der hochinteressanten Einsatzmöglichkeiten ist der Lightpen: Unter Lightpen (oder Lichtgriffel) versteht man einen handlichen Stift, der zur Eingabe oder Bestimmung eines Punktes auf dem Bildschirm dient und den direkten Kontakt zwischen Ihnen und dem Fernseher (Monitor) gestattet. Mit dem Lichtgriffel ist es also möglich, durch ein simples Auflegen der Stiftspitze auf den Bildschirm dem Computer eine Bildschirmposition einzugeben.

Wie geht das nun vonstatten? Sie zeigen mit Ihrem in Controlport 1 gesteckten Lichtgriffel auf einen Punkt des Bildschirms. Dabei ist es egal, ob sich dieser Punkt innerhalb oder außerhalb des eigentlichen Textfensters befindet. Der Computer ist alsdann in der Lage, diesen Punkt zu identifizieren, er kennt also die Koordinaten dieses Punktes. Wenn Sie diese in Ihrem Programm abfragen, können Sie beispielsweise feststellen, ob sich an der Stelle ein bestimmtes Objekt (Buchstabe oder eine Graphik) befindet. Oder Sie zeichnen genau an dieser Stelle einen Punkt in die Graphik, so daß Sie per Hand auf den Bildschirm zeichnen können! Eine andere Idee wäre, den Lightpen als komfortable Cursorsteuerung einzusetzen. Es gibt eine Menge Möglichkeiten der Verwendung.

Doch nun zur technischen Seite des ganzen Geschehens. Wie stellt der Computer fest, wo auf dem Bildschirm nun gerade der Lightpen positioniert ist? Sie wissen, daß ein Bild des Fernsehers (Monitors) aus vielen kleinen Punkten zusammengesetzt ist, die alle 1/20 Sekunden von einem Elektronenstrahl, der auf die erwähnte Lochrasterplatte fällt, zum Aufleuchten gebracht werden. Der Lightpen registriert nun, sofern er auf einen Punkt des Bildschirms gerichtet ist, dieses kurze Aufleuchten und sendet einen Impuls an den Computer. Achten Sie bei der Verwendung des Lightpen deshalb darauf, keine schwarze Hintergrundfarbe zu wählen und den Helligkeitsregler Ihres Bildausgabegerätes nicht zu niedrig einzustellen. Der genannte Impuls erreicht den VIC, der sofort die aktuelle Rasterzeile und die (uns bisher unbekannte) Rasterspalte in zwei Registern als x,y-Punktkoordinaten ablegt. Es sind dies die beiden VIC-Register 19 und 20 (x- und y-Anteil). Hier kann jetzt ein Programm die beiden Werte auslesen und verwerten (z.B. indem es an dieser Stelle einen Punkt zeichnet). Zunächst aber muß noch einiges zu dem Koordinatensystem gesagt werden. Die y-Einteilung, also die Einteilung nach Rasterzeilen kennen Sie bereits. Sie wurde in Kap. 4.7.1 ausführlich erörtert. Die x-Einteilung ist nun etwas komplizierter. Hier gibt es jeweils halb so viele ansprechbare Rasterpunkte, wie wir von der Graphik her kennen, d.h. ein angegebener Rasterpunkt steht für zwei wahre Graphikpunkte. Es ergeben sich folgende Randwerte: Die linke Kante Ihres Bildschirms besitzt etwa den Randwert 30, die rechte ist Spalte Nr. 210. Das reguläre Bildfenster aber liegt links bei ca. 40 und rechts bei 200, womit wir 160 = 320/2 Rasterpunkte im Textfenster besitzen. Angenommen, wir haben zwei Werte für die Rasterkoordinaten aus den Registern 19/20 entnommen und wollen genau an dieser Stelle einen Punkt zeichnen. Dann müssen wir die Rasterkoordinaten in solche für die Graphik umrechnen. Dies können wir nach den obigen Ausführungen durch die folgenden Formeln vornehmen:

```
x = (xp-40)*2
y = yp-40
```

wobei x und y die Graphik- und xr, yr die Rasterkoordinaten darstellen. Jetzt können wir an der errechneten Stelle einen Punkt setzen.

Das ist die einfache und auch von BASIC aus programmierbare Möglichkeit. Doch auch hier bietet Ihnen Ihr Computer eine Möglichkeit, mit der Interrupttechnik zu arbeiten:

Sendet der Lightpen nämlich einen Impuls, so wird das 3. Bit des VIC-Registers 25 gesetzt. Haben wir vorher in Register 26 ebenfalls das 3. Bit gesetzt, kann ein Interrupt ausgelöst werden. Ihr Programm wird also unterbrochen und Ihre Interruptroutine aufgerufen, die das Ereignis bearbeitet. Auch hier werden Programmierbeispiele etc. in Kapitel 5 gegeben.

4.7.3. Sprite-Kollisionen

Wenn Sie sich den Abschnitt 4.5.4.4 durchgelesen haben, wissen Sie bereits, daß Sie evéntuelle Berührungen von Sprites untereinander oder mit Hintergrundzeichen feststellen können, indem Sie den Inhalt der VIC-Register 30 und 31 lesen und analysieren. Wie Sie wissen, ist hier jedem Sprite ein Bit zugeordnet und diejenigen Bits gesetzt, deren Sprite kollidiert ist. Doch es gibt eine weitere Möglichkeit aus Assembler heraus Kollisionen zu registrieren. Wie Sie sich schon denken können, beruht diese Möglichkeit wieder auf der Interrupttechnik. Sie können den VIC (wieder durch den Gebrauch der Register 25/26) veranlassen, bei irgendeiner Kollision einen Interrupt auszulösen. Auch hier werden wieder Berührungen zwischen Sprites und dem Hintergrund und Sprite-Sprite-Kollisionen unterschieden. In Register 25 (IRR) ist dafür jeweils ein Bit reserviert. Bit 3 wird gesetzt, wenn eine im Register 30 näher angegebene Berührung zwischen zwei Sprites stattgefunden hat. Entsprechendes passiert mit Bit 2, falls die Berührung zwischen einem Sprite und einem Hintergrundzeichen stattfand. Ein Interrupt wird, wie Sie sich sicher denken können, aber erst ausgelöst, wenn Sie vorher das korrespondierende Bit in Register 26 gesetzt haben. Vergessen Sie nicht, die IRQ-Adresse umzulegen und nach jedem IRQ das IRR durch Zurückschreiben der Lesedaten zu löschen.

5. Supergraphik intern

Grundsätzliche Graphikprogrammierung in BASIC und Maschinensprache

Nachdem wir jetzt genügend graue Theorie über die Graphikfähigkeiten Ihres Rechners gehört haben, wollen wir uns in diesem Kapitel mit der Realisierung dieser Dinge beschäftigen. Denn was nützt uns das alles, wenn wir nicht wissen, wie wir Sprites oder hochauflösende Graphik selbst einsetzen, ohne auf eine Graphikerweiterung wie die Supergraphik angewiesen zu sein. Es ist das "Know how", das uns fehlt. Aus diesem Grunde gibt es zu jedem der obigen Abschnitte des 4. Kapitels einen dazugehörigen aus diesem Kapitel, der die Programmierung bzw. die Anwendung der einzelnen Möglichkeiten behandelt.

Programme werden möglichst in BASIC, bei den vielen Maschinenspracheroutinen zusätzlich ein BASIC-Lader angegeben. Bei Bedarf werden bereits jetzt Bezüge zum Source-Listing der Supergraphik in Kapitel 6 hergestellt. Alle Programme sind in REM-Zeilen bzw. mit Kommentaren dokumentiert. Die wichtigen und neuen Programmteile werden auch im Text beschrieben. Bei den BASIC-Routinen für die einzelnen Graphikfiguren und -anwendungen muß in aller Deutlichkeit gesagt werden, daß dies selbstverständlich nur Hilfen sind, um das Wesen der einzelnen Vorgänge zu verstehen; die entsprechenden Assemblerroutinen führen die gewünschte Funktion sehr viel schneller aus. Deswegen lohnt sich - wie auch oben schon einmal erwähnt - der Erwerb von Maschinensprachekenntnissen ungeheuer. Wenn Sie BASIC einigermaßen gut beherrschen, ist der Schritt dorthin nicht mehr weit. Versuchen Sie es einmal!

Oft werden Sie sich fragen: Wieso soll ich diese Routine verstehen oder programmieren, ich habe doch die Supergraphik. Aber nur ein paar Befehle wegen gleich die ganze Supergraphik zu laden ist natürlich etwas übertrieben. Aus diesem Grunde geben wir auch noch später viele Lösungsmöglichkeiten vor, Graphik auch ohne Supergraphik zu programmieren.

5.1 Text und Graphik auf dem Low-Res-Bildschirm

Die erste und einfachste Möglichkeit, Graphiken auf Ihrem Bildschirm darzustellen, ist das Arbeiten mit den originalen Graphikzeichen, die Sie an der vorderen Seite der einzelnen Tasten Ihres Rechners finden. Schon hiermit lassen sich mit etwas Phantasie (sie wird bei der Graphikerstellung stets benötigt) wunderbare und vielfarbige Bilder erzeugen. Die Graphikzeichen lassen sich auf verschiedene Art und Weise erreichen:

a) Ansteuerung durch die Tastatur:

Die Zeichen, die auf der rechten Seite der Tasten stehen, lassen sich durch gleichzeitiges Drücken dieser jeweiligen normalen mit der <shift>-Taste auf dem Bildschirm darstellen. Diejenigen, die links stehen, werden mit der gedrückten <C=> (<commodore>) -Taste angewählt. Zusätzlich zu diesen normalen können Sie noch sogenannte inverse Zeichen erzeugen, indem Sie vorher gleichzeitig auf die Tasten <ctrl> und <rvs on> drücken (oder Sie geben ein: PRINT CHR$(18)). Wollen Sie die ausgesuchten Zeichen nicht mehr invers darstellen, so genügt ein <ctrl> mit <rvs off> (PRINT CHR$(146)).

Farben:

Die 16 verschiedenen Zeichen-Farben, die auf dem Bildschirm dargestellt werden können, sind ebenfalls durch die Tastatur anwählbar (eine umfassende Tabelle der Farben und ihrer verschiedenen Zuordnungen finden Sie im Anhang):

Dabei drücken Sie für die ersten 8 Farben (numeriert nach den Farbcodes) gleichzeitig mit der <ctrl>-Taste eine der Tasten <1-8> (die dabei entstehende Farbe steht ebenfalls auf der Frontseite der 8 Tasten). Wollen Sie dagegen den folgenden Zeichen eine der 8 letzten Farben (Farben Nr. 8-15) geben, so drücken Sie einfach die <C=>-Taste mit den erwähnten 8 Farbtasten.

b) Ansteuerung durch das PRINT-Statement:

Jedes Zeichen besitzt einen bestimmten, sogenannten ASCII-Code. Durch Angabe dieses Codes kann jedes Zeichen eindeutig bestimmt werden (dabei besteht zwischen inversen und normalen Zeichen allerdings kein Unterschied). Den ASCII-Code eines Zeichens können Sie durch den BASIC-Befehl ASC in der folgenden Weise ermitteln (In diesem Falle für das Zeichen "A"):

```
PRINT ASC("A")            oder
Z$ = "A" : PRINT ASC(Z$)
```

Der Rechner schreibt: 65. Kennen Sie umgekehrt den ASCII-Code eines Zeichens, so finden Sie dieses wie folgt durch den Befehl CHR$ (ebenfalls für das Zeichen "A"):

```
PRINT CHR$(65)            oder
C = 65 : PRINT CHR$(C)
```

Und schon steht ein A auf dem Bildschirm. Der Vorteil dieser Codierung ist eine Berechenbarkeit von Zeichen, d.h. Sie können ein Zeichen durch irgendeinen Algorithmus (Rechenvorschrift) bestimmen. Gleichzeitig werden Vergleiche o.ä. sehr vereinfacht.

Farben:

Auch die verschiedenen Farben (neben vielen anderen Control-Funktionen wie <clr/home> etc.) besitzen ASCII-Codes. Leider werden im CBM 64-Handbuch nur die der ersten 8 Farben angegeben; hier daher eine vollständige Auflistung:

ASCII	Farbe	ASCII	Farbe
144	schwarz	129	orange
5	weiß	149	braun
28	rot	150	hellrot
159	cyan	151	grau 1
156	violett	152	grau 2
30	grün	153	hellgrün
31	blau	154	hellblau
158	gelb	155	grau 3

c) Ansteuerung durch POKE:

Alle Zeichen, die sich auf dem Bildschirm befinden, werden in einem besonderen Speicher abgelegt: dem Video-RAM (s. Kap. 4.6). Hier müssen natürlich normale und inverse Zeichen unterschieden werden, während Controlzeichen, die nicht auf dem Bildschirm erscheinen, nicht vermerkt werden brauchen. Somit wird bei der Abspeicherung ein anderer Code, der sogenannte Bildschirmcode verwendet (s. Kapitel 4). Wollen Sie also direkt in diesen Speicher POKEn, so müssen Sie sich jenes Codes bedienen. Mit

```
POKE 1024, 1
```

beispielsweise bringen Sie ein A in die linke obere Ecke des Bildschirms. Dieses A ist jedoch noch nicht zu sehen (falls dort nicht zufällig vorher schon ein Zeichen stand). Es fehlt die Farbe, die im sogenannten Farb-RAM abgespeichert wird und z.B. mit

```
POKE 55296,5
```

angewählt wird (die 5 stellt den Farbcode dar, der im Farb-RAM abgespeichert wird). Das Zeichen wird grün (Farbe 5). Diese Zusammenhänge sind ausgiebig in Kapitel 4.6.1 dargelegt.

Auf den folgenden Seiten werden ihnen drei Programme vorgestellt, die alle das gleiche Ergebnis (s. Bild) erbringen, welches jedoch auf drei unterschiedliche Arten entsteht, ohne Rücksicht darauf, daß sich die eine oder andere Methode in diesem Falle so gut wie gar nicht für den demonstrierten Zweck eignet. Sie sollen Ihnen lediglich zeigen, wie die einzelnen Möglichkeiten der Zeichendarstellung in vivo, also direkt im Programm realsiert werden können.

```
100 REM ***************************
110 REM **                       **
120 REM **  LOW-GRAPHIK/PRINT    **
130 REM **                       **
140 REM ***************************
150 REM
160 PRINT CHR$(147) : PRINT : PRINT : REM BILDSCHIRM LOESCHEN/LEERZEILEN
170 PRINT"  \ /                           ": REM M,N
180 PRINT" \ ◠              ⌐             ": REM M,U,I,U,K
190 PRINT"   ◡\                           ": REM J,K,M
200 PRINT" / \           ⌐                ": REM N,M,U,K
210 PRINT"               ▂                ": REM P
220 PRINT"           ▁▁▁┴┴▁▁▁             ": REM P,P,P,L,L,P,P
230 PRINT"          /◠\      \            ": REM N,M,M
240 PRINT"         /   \     ▌            ": REM N,M,L
250 PRINT"        /     \    ▌            ": REM N,M,L
260 PRINT"        ┌─────┐   /             ": REM O,Y,Y,Y,Y,P,N
270 PRINT"        │ ┌┐┌┐│  /              ": REM H,A,S,N,N
280 PRINT"        │ └┘└┘│ /  ┌┐◠          ": REM H,Z,X,N,N,A,S,U,I
290 PRINT"        │ ┌──┐│/ ┌┴┴─\          ": REM H,O,P,N,N,U,E,E,K,J,I
300 PRINT"        │ ┌ ││/ ─┴o──o┴         ": REM H,O,N,N,N,-,E,W,C,C,W,E
310 PRINT"        ─┬────  ┌───────────────": REM Y,Y,O,Y,Y,Y,O,Y,Y,...
320 PRINT"          └────────┘            ": REM L,P,P,P,P,P,@
```

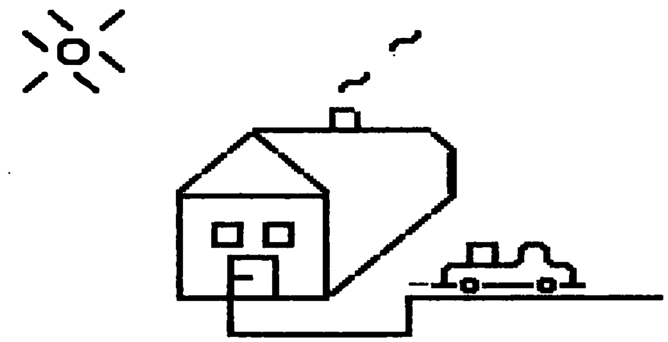
```
100 REM *************************
110 REM **                     **
120 REM **  LOW-GRAPHIK/CHR$   **
130 REM **                     **
140 REM *************************
150 REM
160 PRINT CHR$(147) : PRINT : PRINT : REM BILDSCHIRM LOESCHEN/LEERZEILEN
170 FOR X=1 TO 290 : REM 290 ASCII-CODES EINLADEN
180 READ CH : REM LESE DATA
190 PRINT CHR$(CH); : REM SCHREIBE ZEICHEN
200 NEXT X
210 REM
220 REM *******************
230 REM **  DATAZEILEN  **
240 REM *******************
250 REM
260 DATA  32,  32, 205,  32, 206,  13
270 REM
280 DATA  32, 205, 213, 201,  32,  32
290 DATA  32,  32,  32,  32,  32,  32
300 DATA  32,  32,  32, 213, 203,  13
310 REM
320 DATA  32,  32, 202, 203, 205,  13
330 REM
340 DATA  32, 206,  32, 205,  32,  32
350 DATA  32,  32,  32,  32,  32,  32
360 DATA  32, 213, 203,  13
370 REM
380 DATA  32,  32,  32,  32,  32,  32
390 DATA  32,  32,  32,  32,  32,  32
400 DATA  32, 175,  13
410 REM
420 DATA  32,  32,  32,  32,  32,  32
430 DATA  32,  32,  32,  32, 175, 175
440 DATA 175, 204, 204, 175, 175,  13
450 REM
460 DATA  32,  32,  32,  32,  32,  32
470 DATA  32,  32,  32, 206, 205,  32
480 DATA  32,  32,  32,  32,  32, 205
490 DATA  13
500 REM
510 DATA  32,  32,  32,  32,  32,  32
520 DATA  32,  32, 206,  32,  32, 205
530 DATA  32,  32,  32,  32,  32, 182
540 DATA  13
550 REM
560 DATA  32,  32,  32,  32,  32,  32
570 DATA  32, 206,  32,  32,  32,  32
580 DATA 205,  32,  32,  32,  32, 182
590 DATA  13
```

```
600 REM
610 DATA  32,  32,  32,  32,  32,  32
620 DATA  32, 207, 183, 183, 183, 183
630 DATA 208,  32,  32,  32,  32, 206
640 DATA  13
650 REM
660 DATA  32,  32,  32,  32,  32,  32
670 DATA  32, 165, 176, 174, 176, 174
680 DATA 170,  32,  32,  32, 206,  13
690 REM
700 DATA  32,  32,  32,  32,  32,  32
710 DATA  32, 165, 173, 189, 173, 189
720 DATA 170,  32,  32, 206,  32,  32
730 DATA 176, 174, 213, 201,  13
740 REM
750 DATA  32,  32,  32,  32,  32,  32
760 DATA  32, 165,  32, 207, 208,  32
770 DATA 170,  32, 206,  32,  32, 213
780 DATA 177, 177, 203, 202, 201,  13
790 REM
800 DATA  32,  32,  32,  32,  32,  32
810 DATA  32, 165,  32, 207, 170,  32
820 DATA 170, 206,  32,  32,  45, 177
830 DATA 215, 195, 195, 215, 177,  13
840 REM
850 DATA  32,  32,  32,  32,  32,  32
860 DATA  32, 183, 183, 207, 183, 183
870 DATA 183,  32,  32,  32, 207, 183
880 DATA 183, 183, 183, 183, 183, 183
890 DATA 183, 183,  13
900 REM
910 DATA  32,  32,  32,  32,  32,  32
920 DATA  32,  32,  32, 204, 175, 175
930 DATA 175, 175, 175, 175, 165,  13
```

```
100 REM **************************
110 REM **                      **
120 REM **  LOW-GRAPHIK/POKE     **
130 REM **                      **
140 REM **************************
150 REM
160 FA = 5 : REM FARBE = GRUEN
170 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
180 VR = 1024 + 2*40 : REM POKE-STARTADRESSE (VIDEORAM)
190 FR = 55296 + 2*40 : REM POKE-STARTADRESSE (FARBRAM)
200 FOR Y=0 TO 15 : REM 16 ZEILEN
210 READ ZA : REM ANZAHL DER ZEICHEN IN DER ZEILE HOLEN
220 VR = VR+40 : REM NAECHSTE ZEILE (40 SPEICHERSTELLEN WEITER)
230 FR = FR+40 : REM NAECHSTE ZEILE (40 SPEICHERSTELLEN WEITER)
240 FOR X=0 TO ZA-1 : REM ZA ZEICHEN POKEN
```

```
250 READ BC : REM BILDSCHIRMCODE LESEN
260 POKE VR+X, BC : REM UND IN VIDEORAM SCHREIBEN
270 POKE FR+X, FA : REM FARBE EINPOKEN
280 NEXT X
290 NEXT Y
300 REM
310 REM ************************
320 REM **  BILDSCHIRMCODES  **
330 REM ************************
340 REM
350 DATA   5
360 DATA  32,  32,  77,  32,  78
370 REM
380 DATA  17
390 DATA  32,  77,  85,  73,  32,  32
400 DATA  32,  32,  32,  32,  32,  32
410 DATA  32,  32,  32,  85,  75
420 REM
430 DATA   5
440 DATA  32,  32,  74,  75,  77
450 REM
460 DATA  15
470 DATA  32,  78,  32,  77,  32,  32
480 DATA  32,  32,  32,  32,  32,  32
490 DATA  32,  85,  75
500 REM
510 DATA  14
520 DATA  32,  32,  32,  32,  32,  32
530 DATA  32,  32,  32,  32,  32,  32
540 DATA  32, 111
550 REM
560 DATA  17
570 DATA  32,  32,  32,  32,  32,  32
580 DATA  32,  32,  32,  32, 111, 111
590 DATA 111,  76,  76, 111, 111
600 REM
610 DATA  18
620 DATA  32,  32,  32,  32,  32,  32
630 DATA  32,  32,  32,  78,  77,  32
640 DATA  32,  32,  32,  32,  32,  77
650 REM
660 DATA  18
670 DATA  32,  32,  32,  32,  32,  32
680 DATA  32,  32,  78,  32,  32,  77
690 DATA  32,  32,  32,  32,  32, 118
700 REM
710 DATA  18
720 DATA  32,  32,  32,  32,  32,  32
730 DATA  32,  78,  32,  32,  32,  32
740 DATA  77,  32,  32,  32,  32, 118
750 REM
```

```
760 DATA  18
770 DATA  32,  32,  32,  32,  32,  32
780 DATA  32,  79, 119, 119, 119, 119
790 DATA  80,  32,  32,  32,  32,  78
800 REM
810 DATA  17
820 DATA  32,  32,  32,  32,  32,  32
830 DATA  32, 101, 112, 110, 112, 110
840 DATA 106,  32,  32,  32,  78
850 REM
860 DATA  22
870 DATA  32,  32,  32,  32,  32,  32
880 DATA  32, 101, 109, 125, 109, 125
890 DATA 106,  32,  32,  78,  32,  32
900 DATA 112, 110,  85,  73
910 REM
920 DATA  23
930 DATA  32,  32,  32,  32,  32,  32
940 DATA  32, 101,  32,  79,  80,  32
950 DATA 106,  32,  78,  32,  32,  85
960 DATA 113, 113,  75,  74,  73
970 REM
980 DATA  23
990 DATA  32,  32,  32,  32,  32,  32
1000 DATA  32, 101,  32,  79, 106,  32
1010 DATA 106,  78,  32,  32,  45, 113
1020 DATA  87,  67,  67,  87, 113
1030 REM
1040 DATA  26
1050 DATA  32,  32,  32,  32,  32,  32
1060 DATA  32, 119, 119,  79, 119, 119
1070 DATA 119,  32,  32,  32,  79, 119
1080 DATA 119, 119, 119, 119, 119, 119
1090 DATA 119, 119
1100 REM
1110 DATA  17
1120 DATA  32,  32,  32,  32,  32,  32
1130 DATA  32,  32,  32,  76, 111, 111
1140 DATA 111, 111, 111, 111, 101
```



Das erste der drei Programme zeigt die Erstellung eines Bildes mittelst PRINT-Statement, die hier schnellste und kürzeste und damit in diesem Fall wohl günstigste Möglichkeit. Hier wird das Bild nach dem Löschen des Bildschirmes, was in Zeile 160 durch ein

```
PRINT CHR$(147)
```

geschieht, durch je ein PRINT-Statement pro Zeile zusammengesetzt. Dabei wird ausgiebig von den erwähnten Graphikzeichen Gebrauch gemacht. In den REM-Zeilen dahinter erfahren Sie, auf welchen Tasten Sie die einzelnen Gebilde finden (die Leerzeichen sind dabei natürlich ausgelassen). Sie müssen diese Tasten dann nur noch gemeinsam mit <shift> oder <C=> drücken (wie oben beschrieben), und schon erscheint das gewünschte Zeichen auf Ihrem Bildschirm.

Im zweiten Programm wird das gespeicherte Bild ebenfalls durch PRINT-Statements erzeugt. Sie enthalten jedoch nicht direkt die einzelnen Zeichen, sondern diese werden über den Umweg der ASCII-Codes erzeugt. Die verschiedenen ASCII-Codes sind dabei in DATA-Zeilen untergebracht. Wie Sie vielleicht wissen, werden in DATA-Zeilen verschiedene durch Komma abgetrennte Elemente gespeichert, die dann durch den Befehl READ nacheinander(!) in einen beliebigen Speicher (hier CH) eingelesen werden können (s. CBM 64 - Handbuch Kapitel 8). Dieses Einlesen geschieht in unserem Programm in Zeile 180. Die Variable CH enthält nun den ASCII-Wert des als nächstes auszugebenden Wertes. In Zeile 190 wird das zugeordnete Zeichen dann gePRINTet. Das Ganze spielt sich in einer FOR...NEXT-Schleife ab, die insgesamt 290 mal durchläuft, um alle 290 Daten einzulesen. Am Ende jeder Bildzeile (die in den DATA-Zeilen durch ein REM getrennt sind) steht, wie Sie sehen, die Zahl 13. Dies ist der ASCII-Code für <return> und veranlaßt den Computer das nächste Zeichen an den Anfang der nächsten Zeile zu setzen. Wie Sie weiterhin sehen, werden hier eine ganze Menge DATAs benötigt, was diese Methode in unserem Falle recht uneffektiv gestaltet. Ein wenig geändert wäre die Lage, wenn man statt der vielen Codes für die Leerzeichen (ASCII = 32) am Anfang einer Zeile einen Merker angibt, der über die Anzahl der Leerzeichen Auskunft gibt, die ausgegeben werden müssen, bevor die richtigen Graphikzeichen erscheinen. Lassen Sie sich einmal etwas einfallen.

Beispiel Nr. 3 demonstriert uns die Anwendung des POKE-Befehls, um Zeichen direkt in den Video-RAM einzuschreiben. Auch hier werden diesmal die Bildschirmcodes in DATA-Zeilen untergebracht. Doch neben dem einfachen Einschreiben eines Zeichens in den Speicher müssen Sie weiterhin noch die Farbe jedes einzelnen Buchstaben etc. in den Farb-RAM eintragen, wie oben dargelegt. Kern des Programms sind zwei ineinander verschachtelte FOR...NEXT-Schleifen. Die äußere (Z. 200-290) erhöht (nach dem Durchlauf der inneren) die Nummer der Zeile, die mit Graphikzeichen gefüllt werden soll. Vor dem Start der inneren Schleife wird zunächst ein Wert aus den DATA-Zeilen in die Variable ZA gelesen (Z. 210), der die Anzahl der Zeichen in der jeweiligen Bildschirm-Zeile angibt. Dieser dient dazu die Anzahl der inneren Schleifendurchläufe zu bestimmen. In dem Inneren dieser Schleife werden nun nacheinander die Bildschirmcodes der verschiedenen Zeichen durch READ eingelesen (Z. 250) und an die laufende Adresse im Video-RAM gePOKEt (Z. 260). Alsdann schreiben wir in die korrespondierende Stelle des Farb-RAMs den Wert 5 für die Farbe grün (Z. 270), der in Zeile 160 festgelegt wurde. Auch hier bietet sich natürlich die gleiche Verkürzung der DATA-Zeilen, wie im zweiten Beispiel beschrieben, an.

Wie stellt man aber nun am günstigsten ein eigenes Graphikbild zusammen? Hier gibt es die unterschiedlichsten Möglichkeiten, und jeder wird selbst entscheiden, welche von ihnen ihm am einfachsten erscheinen. Allgemein kann aber gesagt werden, daß ein richtig schönes Bild mit vielen Details eine recht zeitaufwendige Sache ist, besonders, wenn man dazu noch selbstdefinierte Zeichen mit ins Spiel bringt, wie dies in Paragraph 5.4 dargelegt ist. Doch können auch einfache errechnete (also durch eine bestimmte Rechenvorschrift erzeugte) Bilder oft schöne Effekte erzeugen. Dies demonstriert z.B. das folgende Beispiel:

```
100 REM ********************
110 REM **                **
120 REM **  ZUFALLSBILD   **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
170 PRINT CHR$(RND(1)*3 + 177); : REM EINES DER DREI ZEICHEN
ASCII=177/178/179
180 GOTO 170
```

Vielleicht versuchen Sie dieses Programm einmal zu verstehen. Ein Tip: RND(1) ergibt Zufallszahlen von 0-1.

Eine weitere Möglichkeit zur Erstellung von Bildern ist mit der Verwendung einer besonderen Routine verbunden. Diese Routine positioniert den aktuellen Cursor an eine beliebige Stelle des Bildschirms. Wie Sie wissen ist dies mit dem originalen BASIC nur innerhalb einer Zeile möglich. Die kleine Unterroutine ab Zeile 1000 des folgenden Programms aber läßt einen beliebigen Zugriff zu. Sie übernimmt die gleiche Aufgabe wie der POS=-Befehl der Supergraphik:

```
100 REM **************************
110 REM **                      **
120 REM **  SINUSKURVE IM TEXT  **
130 REM **                      **
140 REM **************************
150 REM
160 PRINT CHR$(147) : REM BILDSCHIRM LOESCHEN
170 FOR X=0 TO 39
180 Y=13*SIN(X/3)+12 : REM FUNKTION
190 GOSUB 1000 : PRINT"*"; : REM POSITION BERECHNEN
200 NEXT X : END
210 REM
220 REM POSITIONSBERECHNUNG:
230 REM ********************
1000 PRINT CHR$(19);:IF Y>0 THEN FOR Z=1 TO Y:PRINT:NEXT Z
1010 PRINT TAB(X);: RETURN
```

Gezeichnet wird eine Sinuskurve (s. Kap. 5.1) mit Hilfe des Sternzeichnes (*). Dabei werden dem Unterprogramm in Zeile 1000 in den Speichern X und Y die Parameter für die Spalte (X) und die Reihe (Y) der gewünschten Cursorposition übergeben. Nach einem <home> (PRINT CHR$(19);) werden solange carrige returns gesendet, bis die gewünschte Zeile ereicht ist. Dann muß nur noch durch einen TAB-Befehl die richtige Spalte ausgewählt werden. Dieses kleine, aber äußerst effektive Unterprogramm erlaubt Ihnen wunderschöne Graphiken schon im Textmodus.

Ein weiteres recht schönes und übersichtliches Verfahren der Erstellung von Bildern besteht darin, das gewünschte Bild auf dem Fernseher direkt mit dem Cursor und der Tastatur zu erstellen. Dabei lassen Sie die Farbe am besten zunächst einmal außer betracht. Alsdann schreiben Sie vor jede einzelne Zeile (entweder mit Insert, wobei Sie beachten müssen, daß dadurch eventuell Zeilen unten fortgeschoben werden, oder durch einfaches Überschreiben) zunächst die Zeilen-

nummer, ein PRINT (abkürzbar mit einem Fragezeichen (?)) und ein Anführungszeichen ("), gefolgt schließlich von <return> (Sie brauchen also nicht unbedingt ein zweites Anführungszeichen, wenn die Zeile mit diesem PRINT-Ausdruck endet). Damit haben Sie schon einmal die wesentlichen Bildinhalte gespeichert. Doch die Sache hat einige Haken:

Erstens muß in einer Zeile noch Platz für die Zeilennummer, Frage- und Anführungszeichen sein. Soll dieser Platz ebenfalls genutzt werden, so muß dies nachträglich im Programm geschehen. Vorsicht ist in der letzten Bildschirmzeile geboten. Sollten Sie zufällig mit dem Cursor nach unten über diese hinwegrollen, so wird der gesamte Bildschirm nach oben geschoben und die oberste Zeile verschwindet. Ein weiteres Problem: Sie dürfen in keiner Zeile die letzte Spalte verwenden. In diesem Fall würde eine neue Bildschirmzeile eingeschoben, was zu diversen Schwierigkeiten führt.

Weiterhin können keine inversen Zeichen direkt in ein PRINT-Statement aufgenommen werden. Sollte Ihr Bild solche enthalten, so müssen Sie sie durch folgende Sequenz ersetzen:

<rvs on><....><rvs off>

Mit <....> sind alle hintereinander folgenden (im Programm normalen) Zeichen gemeint, die auf dem Bildschirm invers dargestellt werden sollen.

Drittens können Sie ebenfalls keine Farben direkt in ein PRINT-Statement mit aufnehmen. Dies muß durch den Einbau der entsprechenden Farb-Controlcodes in ein PRINT-Statement nach dem Editieren geschehen. Eine Bemerkung zum Schluß: Die Control-Zeichen werden bekanntlich nur dann richtig in einen PRINT-Ausdruck eingebaut (und nicht sofort ausgeführt), wenn Sie vorher ein Anführungszeichen (") gegeben haben. Dies ist aber bei nachträglichen Einfügungen sehr störend und auch nicht unbedingt notwendig. Sie können nämlich durch Einfügen einer Leerstelle mit <inst> ebenfalls an diese Stelle ein Controlzeichen setzen. Wollen Sie also in dem String "AB" zwischen A und B ein Controlzeichen einfügen, so kann dies einfach durch <inst> <controlzeichen> geschehen.

Wie Sie sehen, ist diese Methode der Bilderzeugung nicht gerade sehr komfortabel, doch für den Anfang oder den Gelegenheitsdesigner akzeptabel. Wollen Sie aber professionell Bilder in größerer Stückzahl und Qualität erzeugen, so sollten Sie sich einen kleinen Bildeditor schreiben, also ein Programm, mit dem Sie einfach durch Bewegen

eines (selbst erzeugten) Cursors ein Bild mit allen Farben und Möglichkeiten erstellen können. Dies ist zugegebenermaßen nicht ganz einfach, aber ein sicher lohnendes Projekt. Diejenigen, die sich für dieses Thema besonders interessieren, sollten hierzu unbedingt noch den Paragraphen 5.4 gelesen haben.

5.2 Programmierung der Punktgraphik

Besonders die komplizierte Organisation der hochauflösenden oder gar der Multicolor-Graphik macht einem zu schaffen, wenn man versucht, diese zu bedienen. Allein die Ansteuerung eines Punktes in einem (gedachten) Koordinatensystem verursacht schon recht großes Kopfzerbrechen und schließlich einen enormen Rechenaufwand. Die bloße Berücksichtigung aller notwendigen Faktoren zum Einschalten der Graphik bedarf eines guten Überblicks, der sich erst nach einiger Beschäftigung mit dem Thema Graphik einstellt. Die Erstellung von einfachen Linien oder sogar Kreisen ist dabei so schwer und bedarf vieler mathematischer Kenntnisse, daß sie schon nur noch von recht firmen Programmierern gelöst werden können (falls Sie nicht die Supergraphik verwenden). Aus diesem Grunde werden hier die verschiedenen Routinen (also Programmteile) vorgestellt, die zur Realisierung der in Kapitel 4 dargelegten Möglichkeiten Ihres Rechners notwendig sind. Es ist dabei nicht unbedingt erforderlich, daß jeder einzelne Schritt eines Programms verstanden ist, da letztendlich die Anwendung dieser Dinge ausschlaggebend ist. Wer also nicht weiß, was beispielsweise sin oder cos bedeuten, der sollte über die einzelnen Stellen (hier der Kreiserzeugung) hinweglesen. Trotzdem sollte er den entsprechenden Abschnitt in Kapitel 4 (Abschnitt 4.4) über die Grundlagen der Graphik gelesen haben. Für die Interessierten jedoch können solche Informationen wertvoll sein, um die einzelnen Routinen für eigene Zwecke abzuwandeln oder Teile daraus für ähnliche Aufgaben zu verwenden.

Sie sollten sich jedoch darüber im klaren sein, daß eine BASIC-Routine, so übersichtlich sie sein mag, bei weitem nicht die Geschwindigkeit besitzt, wie ein entsprechendes Maschinenspracheprogramm. Damit sind viele Effekte allein in BASIC nur sehr träge zu verwirklichen. Wenn Sie sich einmal anschauen, wie lange es in BASIC dauert, einen Kreis zu zeichnen, so werden Sie mir da wohl in aller Entschiedenheit und ohne zu zögern zustimmen. Um dieses Manko zu eliminieren, werden wir Ihnen am Ende dieses 5. Kapitels ein kleines Assembler-Graphik-Aid (samt BASIC-Lader) zusammenstellen, das Ihnen komprimiert die Möglichkeiten verschafft, die die Kernpunkte jeder Graphik darstellen. Die einzelnen Funktionen des Graphik-

Paketes können Sie -quasi als kleine BASIC-Erweiterung- von BASIC aus ansteuern. Wollen Sie nur in BASIC programmieren, so beherzigen Sie die Tips, die Ihnen im Anhang zur Optimierung Ihrer Programme gegeben wurden. Mit diesem Graphik-Paket ersparen Sie sich das vorherige Laden der Supergraphik, da Sie die einzelnen Routinen direkt in Ihre Programme einbauen können.

Bei allen Programmen wird davon ausgegangen, daß der Graphikspeicher bei $2000-$3FFF (8192-16383) und der Video-RAM weiterhin bei $0400-$07FF (1024-2047) liegen. Dies macht zwar ein Arbeiten mit Text und Graphik zugleich unmöglich, ist jedoch programmtechnisch besser zu bewältigen. Wie Sie den Graphikspeicher woanders hinlegen können (z.B. unter den ROM) und wie Sie damit umgehen, können Sie dem Source-Listing der Supergraphik entnehmen. Achten Sie aber bei langen Programmen und/oder vielen Speichern darauf, daß diese nicht mit der Graphikseite kollidieren. Sollte dies einmal geschehen, so setzen Sie in der ersten Zeile Ihres Programms einfach durch

```
POKE 45,0 : POKE 46,64
```

den Start der Variablen hoch auf $4000 (16384). Dabei ist jedoch zu beachten, daß bei jeder Programmveränderung die Graphikseite zerstört wird und bei einem Abspeichern auf Diskette oder Kassette die Graphik mit übertragen wird. Wollen Sie also Veränderungen an Ihrem Programm vornehmen nachdem es einmal gestartet worden ist, so müssen Sie es erst einmal wieder einladen und direkt nach der Veränderung abspeichern, bevor Sie es wieder starten!

Wichtig bei allen Angaben ist, daß Sie diese direkt am Computer ausprobieren. Nur so werden Sie Herr über die Unmasse an Fakten und Zusammenhängen und nur so lernen Sie damit umzugehen. Der Computer ist Praxis!

Doch jetzt wollen wir endlich anfangen. Krempeln wir uns also die Ärmel hoch, spucken dreimal in die Hände und los geht's!

5.2.1 Initialisierung der Graphik

Bevor wir unsere Figuren auf den Bildschirm zaubern, müssen wir natürlich erst einmal dafür sorgen, daß überhaupt Graphik zu sehen ist. Dazu gehört, daß der Bildschirm zunächst gelöscht wird, da im Anfang stets einiges an "Müll" erscheinen wird. Der Graphikspeicher wurde schließlich vorher anderweitig genutzt. Zu dem Löschen des Graphikbildschirms gehört natürlich auch ein Löschen der Farbe bzw. das Herstellen eines einfarbigen Bildschirms. Anschließend wollen wir sicher wieder zurück, um Text anzuzeigen. Wir benötigen also insgesamt vier getrennte Programmteile, sogenannte Routinen, allein um in die Graphik einzusteigen:

- Graphik einschalten
- Graphik löschen
- Farbe löschen
- Graphik ausschalten

Diese vier Rechenvorschriften (Algorithmen) werden im folgenden einzeln vorgestellt und besprochen. Sie werden sehen, daß sie in jedem späteren Programm, das sich mit der Graphik beschäftigt, wieder in Form von Unterprogrammen auftauchen werden. Sie sollten also zu Ihrem ständigen Repertoire gehören.

5.2.1.1 Einschalten der Graphik

Nun ist es also soweit, wir können beginnen. Stellen wir zunächst einmal die Dinge zusammen, die zum Einschalten benötigt werden:

a) Speicherlage:

Zunächst einmal müssen wir uns einigen, wo im gesamten Speicherbereich des 64ers die einzelnen Funktionen wie Video-RAM und Graphikspeicher liegen sollen. Hierfür sind Register 24 (Bits 3 und 4-7) des Videocontrollers und das Register 0 (Bits 0 und 1) der CIA 2 zuständig, deren Funktionen ausgiebig in dem Abschnitt 4.3 erläutert werden. In allen unseren Anwendungen werden wir uns - wie oben schon erwähnt - mit dem Graphikspeicher nur in dem Bereich von $2000 bis $3FFF (8192-16383) aufhalten, der Video-RAM liegt bei $0400-$07FF (1024-2047). Wollen Sie Ihre Graphiken in anderen Bereichen ablegen, so müssen Sie entsprechende Änderungen vornehmen.

b) Graphikart:

Wir müssen uns entscheiden, ob wir unser Bild in Multicolor, die bekanntlich eine höhere Farbauflösung zuläßt, dafür aber weniger Punkte in x-Richtung besitzt, oder ob wir die hochauflösende Graphik wählen mit nur einer Farbe pro 8x8-Punkte-Feld. Der Aufbau und die Unterschiede dieser beiden Graphikarten wurden bereits in Paragraph 4.4 ausgiebig erörtert.

Der zweite Punkt ist schnell gelöst. Wir wollen uns mit der hochauflösenden Graphik beschäftigen. Diese wird durch Setzen der Bits 5 und 6 (Bit 6 muß gleichfalls gesetzt werden!) von Register 17 des VIC und das Löschen von Bit 4 des Registers 22 eingeschaltet. Letzteres ist normalerweise gelöscht, braucht also nicht unbedingt gleich Null gesetzt werden. Das alles passiert durch zwei einfache POKEs, die in der unten folgenden Routine in den Zeilen 10070 und 10080 stehen.

Auch die Adreßlagenwahl ist in unserem Falle recht einfach. Da wir uns nicht aus dem unteren 16 K-Bereich unseres Speichers herausbewegen (Sie wissen, daß der VIC nur 16 K addressieren kann (s. Kap. 4.3.2) und im Normalzustand die untersten 16 K für ihn erreichbar sind), brauchen wir keine Veränderungen im Register der CIA 2 zu unternehmen. Lediglich die Basisadresse der Graphikseite muß durch Setzen des 3. Bits von Register 24 in den oberen Teil der 16 K, also nach unseren $2000 (8192) gelegt werden. In dem folgenden Unterprogramm wird dies in Zeile 10090 erreicht:

```
10000 REM ****************************
10010 REM **                        **
10020 REM **  GRAPHIK EINSCHALTEN   **
10030 REM **                        **
10040 REM ****************************
10050 REM
10060 V = 53248 : REM BASISADRESSE - VIDEOCONTROLLER
10070 POKE V+17, PEEK(V+17) OR (8+3)*16 : REM GRAPHIK EIN
10080 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10090 POKE V+24, PEEK(V+24) OR 8 : REM GRAPHIK NACH $2000 (8192)
```

Die Befehle AND und OR sind im Anhang beschrieben. Die Zeilennummern sind extra hoch, um die Routine leicht an ein Programm anzuhängen. Der Textmodus kann auch durch <run/stop> <restore> wieder eingeschaltet werden.

Die Routinen zum Ein- und Ausschalten der Graphik finden Sie im Supergraphik-Listing beim Befehl GMODE, der bekanntlich diese Funktionen übernimmt. Dort können Sie im Listing nachvollziehen, wie diese Aufgabe in der Supergraphik gelöst wurde. Dabei sind natürlich noch eine ganze Menge Faktoren mehr zu beachten. Aus diesem Grunde sind die zuständigen Routinen weitaus komplizierter als hier im "reinen".

5.2.1.2 Löschen der Graphik

Da in dem Speicher, den nun unsere Graphik einnimmt, vorher stets etwas anderes stand, erhalten wir nach dem Einschalten der Graphik ein recht wildes Durcheinander von Strichen oder sonstigen Punkten. Um nun jeden Graphikpunkt zu löschen, müssen wir jedes Bit des Graphikspeichers auf 0 setzen. Mit einem POKE sind wir in der glücklichen Lage, gleich 8 Bits (Byte), also 8 Punkte gleichzeitig anzusprechen. Es genügt also eine FOR...NEXT-Schleife, in der - angefangen von der Graphik-Startadresse ($2000 = 8192) bis zum Ende bei $3FFF (16383) - alle Bytes gelöscht werden. Da nun ein Bild aber nur 320x200 = 64000 Punkte, also 64000/8 = 8000 Bytes besitzt, brauchen wir tatsächlich nur 8000 Bytes zu löschen:

```
10200 REM *************************
10210 REM **                     **
10220 REM **  GRAPHIK LOESCHEN   **
10230 REM **                     **
10240 REM *************************
10250 REM
10260 BG = 8192 : REM BASISADRESSE DES GRAPHIKSPEICHERS
10270 FOR X=BG TO BG+8000 : REM 8000 BYTES
10280 POKE X,0 : REM LOESCHEN
10290 NEXT X
```

Wie Sie sehen, dauert dieser Vorgang recht lange. Haben Sie sich einmal das Graphik-Packet am Ende des Kapitels abgeschrieben, so werden Sie sehen, wie schnell so etwas in Assembler gehen kann. Die Variable BG gehört streng genommen nicht zu der Routine, genauso, wie die Variable V im oberen Programm. Sie sollten am Anfang eines jeden Programmes gesetzt werden. Wollen Sie die einzelnen Routinen als Unterprogramme laufen lassen, so müssen sie mit dem Befehl RETURN abgeschlossen werden.

Im Supergraphik-Listing finden Sie die Routine zum Löschen der Graphik beim GCLEAR-Befehl. Dabei müssen Sie allerdings die verschiedenen Verzweigungen zum Löschen von Graphikfenstern überschlagen. Interessant ist hier nur die Routine zum Löschen eines vollen Bildschirms.

5.2.1.3 Löschen der Farbe

Die Farbe liegt bei der hochauflösenden Graphik stets im Video-RAM (s. Kap. 4.4). Dabei bestimmen die obersten 4 Bits eines jeden Bytes die Farbe der gesetzten Punkte in der Graphikseite, die unteren 4 dagegen die Farbe der nicht gesetzten Punkte, also quasi die der Hintergrundfarbe. Da der Video-RAM vor dem Einschalten der Graphik den Text enthielt, zeigen sich auch nach dem Löschen noch kleine Farbquadrate an den Stellen, an denen vorher Text stand. Um auch diese zu eliminieren, müssen wir im Video-RAM die Farbe einheitlich setzen. Dies wird in der folgenden Routine vorgenommen:

```
10400 REM  **********************
10410 REM  **                  **
10420 REM  **  FARBE LOESCHEN  **
10430 REM  **                  **
10440 REM  **********************
10450 REM
10460 BF = 1024 : REM BASISADRESSE DES VIDEORAM
10470 FA = 6*16 + 7 : REM PUNKT-FARBE=BLAU/HINTERGRUND=GELB
10480 FOR X=BF TO BF+1000 : REM 1000 BYTES
10490 POKE X, FA : REM MIT PUNKT- UND HINTERGRUNDFARBE
10500 NEXT X
```

Hier gilt natürlich das Gleiche bezüglich der Variablen BF, wie im vorigen Programm dargelegt. FA ist ebenfalls eine Variable, die der Routine vom übergeordneten Programm übergeben wird und den Wert enthält, der in jedes Byte des Video-RAMs geschrieben werden soll und damit Punkt- und Hintergrundfarbe bestimmt. Sie sollten (besonders hier) ein wenig an den Programmen verändern, um sie richtig zu verstehen. Dies allerdings muß mit der nötigen Vorsicht geschehen, da wir uns direkt im Herz des Rechners befinden. Lassen Sie beispielsweise BF gleich 0 werden, so wird Ihr Computer nicht zögern, sich von Ihnen zu verabschieden, da Sie direkt die Null-Seite des Speichers manipulieren, das "Kurzzeitgedächtnis" des Betriebssystems.

Wie Sie sich wahrscheinlich denken können, stehen die Routinen zur Farb-Initialisierung bei den Befehlen SCOL=, COLOR= und PCOL= im Source-Listing der Supergraphik.

5.2.1.4 Ausschalten der Graphik

Bislang konnten Sie sich immer nur durch <run/stop> <restore> aus der Graphikanzeige retten. Doch wird dadurch zwangsläufig Ihr Programm beendet, was nicht unbedingt im Sinne des Erfinders ist. Um diese Funktion regulär in unsere Routinensammlung aufzunehmen, müssen wir sämtliche Veränderungen rückgängig machen, die wir in dem Teil "Graphik einschalten" unternommen haben:

- Bits 5/6 - Register 17 löschen
- Bit 4 - Register 22 löschen
- Bit 3 - Register 24 löschen

Dies geschieht im folgenden Programm:

```
10600 REM ****************************
10610 REM **                        **
10620 REM **  GRAPHIK AUSSCHALTEN   **
10630 REM **                        **
10640 REM ****************************
10650 REM
10660 V = 53248 : REM BASISADRESSE - VIDEOCONTROLLER
10670 POKE V+17, PEEK(V+17) AND 255-6*16 : REM GRAPHIK AUS
10680 POKE V+22, PEEK(V+22) AND 255-1*16 : REM MULTICOLOR AUS
10690 POKE V+24, PEEK(V+24) AND 255-8 : REM ZEICHENSATZ WIEDER NAC H
$1000 (4096)
```

Damit haben wir alle wichtigsten Dinge, um die Graphik zu bedienen. Nun können wir uns den schwierigeren Zusammenhängen widmen, die uns ermöglichen, auch etwas auf unserem Bild darzustellen.

5.2.2 Einfache Figuren in der Punktgraphik

Nachdem wir uns mit den Dingen beschäftigt haben, die wir zum Ein- und Ausschalten der Graphik benötigen, kommen wir nun zu den ersten Gehversuchen der Graphikprogrammierung. Angefangen mit der Darstellung eines einfachen Punktes auf dem Bildschirm gehen wir über zu den geometrischen Grundformen der Linie und des Kreises, aus denen näherungsweise fast alle anderen Figuren hergestellt werden können.

5.2.2.1 Punkt

Wir wollen, wie an anderer Stelle schon des öfteren erwähnt, das gesamte Graphikfeld in sogenannte Koordinaten unterteilen. Dabei stellt der erste Wert stets die x-Koordinate, also die Anzahl der Punkte zwischen dem jeweiligen Punkt und dem linken Bildschirmfensterrand (0-319). Der zweite genannte Wert ist dann der y-Anteil der Koordinate, also die Anzahl der Punkte zwischen dem Punkt und der oberen Bildschirmkante (0-199). Der Nullpunkt (Koordinaten: 0,0) liegt demnach in der oberen linken Ecke des Fensters. Die untere rechte Ecke dagegen besitzt die Koordinaten 319,199.

Soweit, sogut. Doch dies ist unsere Vereinbarung. Wir können dem Computer selbst nicht die entsprechenden Koordinaten angeben, um einen Punkt zu bestimmen. Wenn Sie die entsprechenden Kapitel gelesen haben (Kap. 4.4.2.), so kennen Sie den Aufbau der hochauflösenden Graphik und ihre Speicherorganisation. Um nun aus den angegebenen Koordinaten auf das Byte und das Bit zu schließen, das den betreffenden Punkt bestimmt, müssen wir zunächst einige Umrechenarbeit leisten. Sie brauchen die folgenden Ausführungen nicht unbedingt zu verstehen. Den Interessierten unter Ihnen sei die Herleitung der im folgenden Programm verwendeten Formel dargelegt.

Lassen Sie uns zunächst einmal den Einfluß der y-Koordinate untersuchen:

Um die Nummer der Graphikzeile (eine Zeile besteht aus 8 Reihen) zu berechnen, in der sich der Punkt befindet, müssen wir die y-Koordinate lediglich durch 8 teilen (ohne Rest):

zeilennummer = INT(yK/8)

Da jede Zeile aus 320 Bytes besteht (jede Reihe besteht aus 320/8 = 40 Byte), müssen wir diese Nummer mal 320 nehmen, um die Startadresse der betreffenden Zeile relativ zur Startadresse des Graphikspeichers zu erhalten:

zeilenadresse = 320 * INT(yK/8)

Der Rest der eben durchgeführten Division stellt nun die Nummer der Reihe in dieser Zeile dar und muß nur noch hinzuaddiert werden:

reihenadresse = 320 * INT(yK/8) + (y AND 7)

Der Einfluß der x-Koordinate ist etwas schwieriger, da hier nicht nur einzelne Bytes, sondern sogar die Bits unterschieden werden müssen: Als erstes berechnen wir die Adresse des angesprochenen Bytes relativ zur Startadresse der betreffenden Reihe (s.o.). Wir rechnen:

byteadresse = 8 * INT(xK/8)

Nun berechnen wir die Position des gewünschten Bits in dem betreffenden Byte durch Erstellung einer Maske. Das jeweilige Bit wird in der Maske gesetzt, alle anderen sind gleich 0:

maske = 2^(7-(xK AND 7))

Diese Einzelteile werden - wie in der folgenden Routine gezeigt - zusammengesetzt:

```
10700 REM ************************
10710 REM **                    **
10720 REM **  PUNKTBERECHNUNG   **
10730 REM **     (SETZEN)       **
10740 REM ************************
10750 REM
10760 RA = 320 * INT(YK/8) + (YK AND 7)
10770 BA = 8 * INT(XK/8)
10780 MA = 2^(7-(XK AND 7))
10790 AD = SA + RA + BA
10800 POKE AD, PEEK(AD) OR MA
10810 REMzfp7
10900 REM ************************
10910 REM **                    **
10920 REM **  PUNKTBERECHNUNG   **
10930 REM **    (LOESCHEN)      **
10940 REM ************************
10950 REM
```

```
10960 RA = 320 * INT(YK/8) + (YK AND 7)
10970 BA = 8 * INT(XK/8)
10980 MA = 255 - 2^(7-(XK AND 7))
10990 AD = SA + RA + BA
11000 POKE AD, PEEK(AD) AND MA
11010 REMzfp3
11020 REM INTERNE PARAMETER:
11030 REM ******************
11040 REM RA: REIHENADRESSE
11050 REM BA: BYTEADRESSE
11060 REM MA: MASKE
11070 REM AD: ZIELADRESSE
11080 REMzfp3
11090 REM VORZUGEBENDE PARAMETER:
11100 REM ***********************
11110 REM SA: GRAPHIKSPEICHERSTARTADRESSE (Z.B. 8192)
11120 REM XK: X-KOORDINATE
11130 REM YK: Y-KOORDINATE
```

Wie Sie sehen, unterscheiden wir hier zwischen dem Setzen und dem Löschen eines Punktes in HGR. Tatsächlich müssen diese Fälle getrennt behandelt werden. Die Variable SA gibt die Anfangsadresse des betreffenden Graphikspeichers an und wird bei uns stets bei 8192 gehalten. Natürlich werden die beiden Routinen wie die im letzten Abschnitt vorgeführten meist als Unterprogramme verwendet und enden daher zum größten Teil mit einem RETURN. Dies erkennen Sie bereits in dem folgenden Programm, das die inzwischen vorgeführten Routinen anwendet:

```
100 REM    ******************
110 REM    **              **
120 REM    **  SINUSKURVE  **
130 REM    **              **
140 REM    ******************
150 REM
160 V=53248 : REM STARTADRESSE DES VIC
170 SA = 8192 : REM STARTADRESSE DES GRAPHIKSPEICHERS
175 POKE V+32, 10 : REM RAHMENFARBE
180 GOSUB 10000 : REM GRAPHIK EINSCHALTEN
190 GOSUB 10200 : REM GRAPHIK LOESCHEN
200 FA = 7*16 + 2 : GOSUB 10400 : REM FARBE SETZEN
210 YK = 100 : REM X-ACHSE ZEICHNEN
220  FOR XK=0 TO 319
230   GOSUB 10700 : REM PUNKT ZEICHNEN
240  NEXT XK
250 XK = 160 : REM Y-ACHSE ZEICHNEN
260  FOR YK=0 TO 199
```

```
270  GOSUB 10700 : REM PUNKT ZEICHNEN
280 NEXT YK
290 FOR XK=0 TO 319 : REM SINUSKURVE ZEICHNEN
300  YK = 70 * SIN (XK/25.5) + 99
310  GOSUB 10700 : REM PUNKT ZEICHNEN
320 NEXT XK
330 POKE 198,0 : REM TASTEN LOESCHEN
340 WAIT 198,255 : REM AUF TASTE WARTEN
350 GOSUB 10600 : REM GRAPHIK AUS
360 END
370 REM
10000 REM ***************************
10010 REM **                       **
10020 REM **  GRAPHIK EINSCHALTEN  **
10030 REM **                       **
10040 REM ***************************
10050 REM
10070 POKE V+17, PEEK(V+17) OR (8+3)*16 : REM GRAPHIK EIN
10080 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10090 POKE V+24, PEEK(V+24) OR 8 : REM GRAPHIK NACH $2000 (8192)
10100 RETURN
10110 REM
10200 REM  ***********************
10210 REM  **                   **
10220 REM  **  GRAPHIK LOESCHEN  **
10230 REM  **                   **
10240 REM  ***********************
10250 REM
10270 FOR X=SA TO SA+8000 : POKE X,0 : NEXT X
10300 RETURN
10310 REM
10400 REM   *********************
10410 REM   **                 **
10420 REM   **  FARBE LOESCHEN  **
10430 REM   **                 **
10440 REM   *********************
10450 REM
10460 BF = 1024 : REM BASISADRESSE DES VIDEORAM
10480 FOR X=BF TO BF+1000 : POKE X,FA : NEXT X
10510 RETURN
10520 REM
10600 REM ***************************
10610 REM **                       **
10620 REM **  GRAPHIK AUSSCHALTEN  **
10630 REM **                       **
10640 REM ***************************
10650 REM
10670 POKE V+17, PEEK(V+17) AND 255-6*16 : REM GRAPHIK AUS
10680 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10690 POKE V+24, PEEK(V+24) AND 255-8 : REM ZEICHENSATZ WIEDER NAC H
$1000 (4096)
```

```
10695 RETURN
10700 REM ************************
10710 REM **                    **
10720 REM **  PUNKTBERECHNUNG   **
10730 REM **      (SETZEN)      **
10740 REM ************************
10750 REM
10760 RA = 320 * INT(YK/8) + (YK AND 7)
10770 BA = 8 * INT(XK/8)
10780 MA = 2^(7-(XK AND 7))
10790 AD = SA + RA + BA
10800 POKE AD, PEEK(AD) OR MA
10810 RETURN
```

Bis auf ein paar Änderungen (z.B. wurde in dem Unterprogramm "Graphik löschen", aus Geschwindigkeitsgründen weitestgehendst auf REM-Zeilen verzichtet). Sind die verwendeten Routinen identisch mit den bisher vorgestellten. Probieren Sie ruhig einmal die einzelnen Dinge aus (besonders in der Zeile 300 sollten Sie die verschiedenen Zahlen verändern). Nur so lernen Sie mit ihnen umzugehen.

Im Supergraphik-Listing finden Sie die Routine zur Berechnung der Adresse eines Punktes unter dem Label HPOSN. Sie macht genau das, was wir oben besprochen haben. Die zentrale Routine, die einen Punkt in die Graphik setzt finden Sie unter dem Label PLT. Dieser Routine werden aber noch einige Flags (Zeichenflag in FLG und FLG2), die Adresse in ADL/ADH, das Graphikmodusflag in GFLAG usw. übergeben.

5.2.2.2 Linie

Schon etwas schwieriger gestaltet sich das Zeichnen einer Linie zwischen zwei beliebigen Punkten auf dem Bildschirm. Man sieht dies zwar täglich in irgendwelchen Programmdemos, macht sich jedoch nie richtig Gedanken darüber, welche Überlegungen dahinter stecken. Das Problem ist: wie stelle ich fest, welche Punkte des Bildschirms auf dieser Linie liegen. Um es zu lösen müssen wir uns ein wenig mit der sogenannten analytischen Geometrie beschäftigen. Bekommen Sie keinen Schreck! Hinter diesem monströsen Begriff verbirgt sich etwas ganz harmloses (jedenfalls in dem Rahmen, der uns hier interessiert) und wenn es Sie nicht so sehr interessieren sollte, etwa weil sich Ihnen damit üble Kindheitserinnerungen verbinden, dann können Sie die folgenden Zeilen ruhig überlesen. Was wir suchen ist eine Formel, mit

der wir die Punkte einer Geraden berechnen können, deren Eckpunkte gegeben sind.

Nahezu jeder von uns wird schon einmal in irgendeinem Zusammenhang (meist aus der Schule her) von der sogenannten normierten Geradengleichung gehört haben:

$$y = mx + n$$

wobei x und y die Koordinaten eines Punktes auf einer Geraden, m die Steigung der Geraden und n den Schnittpunkt mit der y-Achse darstellen. Durch einfache Umformung dieser Formel erhalten wir:

$$n = y - mx$$

Kennen wir nun zwei Punkte der Geraden (unsere Endpunkte x1,y1 und x2,y2), so können wir gleichfalls die zwei folgenden Formeln aufsetzen, die wir gleichsetzen können:

$$n = y_1 - mx_1 \quad ---- \quad n = y_2 - mx_2$$

$$\rightarrow \quad y_1 - mx_1 = y_2 - mx_2$$

$$\Leftrightarrow \qquad m = \frac{y_2 - y_1}{x_2 - x_1}$$

Letztere Formel läßt uns nun die Steigung m der obigen Gleichung ausrechnen. Ist n=0, so geht die Gerade durch den Ursprung mit Koordinaten 0,0. Verschieben wir diesen Ursprung der Gerade zu einem Endpunkt, so müssen wir entsprechend die beiden Koordinaten (in diesem Fall x2 und y2) zu x und y hinzuaddieren. Die folgende Formel gibt uns nun die endgültige Geradengleichung wieder, die bereits die verschobene Gerade angibt und in die m eingesetzt wurde:

$$y = \frac{y_2 - y_1}{x_2 - x_1} * (x - x_2) + y_2$$

Diese Formel ist die Grundlage des unten dargestellten Programms und wird stückweise in den Zeilen 10970, 11000 und 11020 errechnet, wobei die x-Koordinate XK stets von X2 nach X1 läuft und für jeden solchen x-Wert der entsprechende y-Wert bestimmt wird. Ein Schaubild mag diese Formel erläutern:

Das einzige Problem bei dieser Formel entsteht, wenn wir eine Senkrechte zeichnen wollen. In diesem Fall wird x1=x2 und damit der Nenner der Steigung gleich 0, was zu einem DIVISION BY ZERO ERROR führt. Wir umgehen diese Unkorrektheit, indem wir in Zeile 10990 verzweigen und dort direkt eine Senkrechte zeichnen. Wie Sie sehen werden und was schon oft erwähnt wurde, kommt ein BASIC-Programm in der Geschwindigkeit mit einem Maschinenprogramm natürlich nicht mit. Trotzdem mag Ihnen diese Routine, die Sie ebenfalls als Unterprogramm verwenden können, gute Dienste leisten.

```
100 REM        ***************
110 REM        **           **
120 REM        **  GERADE   **
130 REM        **           **
140 REM        ***************
150 REM
160 V=53248 : SA=8192
170 GOSUB 10000 : REM GRAPHIK EIN
180 FA = 1*16 + 0 : GOSUB 10400 : REM FARBE SETZEN
190 GOSUB 10200 : REM GRAPHIK LOESCHEN
270 X1=110:Y1=120:X2=130:Y2=140 :REM ENDPUNKT-KOORDINATEN
280 GOSUB 10900 : REM GERADE
290 WAIT 198,255 : REM AUF TASTE WARTEN
300 GOSUB 10600 : REM GRAPHIK AUS
310 END
320 REM
10000 REM ****************************
10020 REM **  GRAPHIK EINSCHALTEN   **
10040 REM ****************************
10050 REM
10070 POKE V+17, PEEK(V+17) OR (8+3)*16 : REM GRAPHIK EIN
10080 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10090 POKE V+24, PEEK(V+24) OR 8 : REM GRAPHIK NACH $2000 (8192)
10100 RETURN
10110 REM
10200 REM  ************************
10220 REM  **  GRAPHIK LOESCHEN  **
10240 REM  ************************
10250 REM
10270 FOR X=SA TO SA+8000 : POKE X,0 : NEXT X
10300 RETURN
10310 REM
10400 REM   **********************
10420 REM   **  FARBE LOESCHEN  **
10440 REM   **********************
10450 REM
10460 BF = 1024 : REM BASISADRESSE DES VIDEORAM
```

```
10480 FOR X=BF TO BF+1000 : POKE X,FA : NEXT X
10510 RETURN
10520 REM
10600 REM ****************************
10620 REM **  GRAPHIK AUSSCHALTEN  **
10640 REM ****************************
10650 REM
10670 POKE V+17, PEEK(V+17) AND 255-6*16 : REM GRAPHIK AUS
10680 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10690 POKE V+24, PEEK(V+24) AND 255-8 : REM ZEICHENSATZ WIEDER NAC H
$1000 (4096)
10695 RETURN
10700 REM ***********************
10720 REM **  PUNKTBERECHNUNG  **
10730 REM **     (SETZEN)     **
10740 REM ***********************
10750 REM
10760 RA = 320 * INT(YK/8) + (YK AND 7)
10770 BA = 8 * INT(XK/8)
10780 MA = 2^(7-(XK AND 7))
10790 AD = SA + RA + BA
10800 POKE AD, PEEK(AD) OR MA
10810 RETURN
10900 REM
10910 REM ***********************
10930 REM **  GERADE ZEICHNEN  **
10950 REM ***********************
10960 REM
10970 DY=Y2-Y1:DX=X2-X1 :REM DIFFERENZEN
10980 YK=Y2:XK=X2 :REM Y-START
10990 IFDX=0THEN FOR YK=Y2TOY1 STEPSGN(-DY):GOSUB 11060:NEXT YK:GOTO
11050 :REM SENKR.
11000 DD=DY/DX :REM STEIGUNG
11010 FOR XK=X2 TO X1 STEP SGN(-DX)
11020 ZK=INT(DD*(XK-X2)+Y2) :REM GERADENGLEICHUNG
11030 IF ZK<>YK THEN YK=YK+SGN(-DY):GOSUB 11060:GOTO 11030 :REM SE NKR.
ZEICHNEN
11040 GOSUB11060:NEXT XK: REM NAECHSTE X-KOORD.
11050 RETURN
11060 GOSUB10760:XK=XK+1:GOSUB10760:XK=XK-1:RETURN :REM DOPPELT BR EIT
ZEICHNEN
```

Sollten Sie es einmal leid sein, stets darauf zu warten, bis der gesamte Bildschirm gelöscht ist, so setzen Sie einfach vor die Zeile 190 ein REM, um diese Prozedur zu unterdrücken.

In der obigen Routine werden einige Speicher verwendet, deren Inhalt im folgenden kurz erläutert sei:

Eingabewerte:
X1/Y1 bzw.
X2/Y2: Endkoordinaten der Linie

interne Werte:
DX/DY: Differenzen der Koordinatenpaare
DD: Die Steigung m
XK/YK: Koordinaten des aktuellen Punktes
ZK: Zwischenspeicher

Zwei Punkte müssen hier noch erläutert werden: Zum einen die Funktion SGN, zum anderen die Zeile 11060. SGN besitzt eine recht nützliche Eigenschaft: Ist die Zahl, die in den Klammern steht positiv, so ist das Ergebnis 1, ist sie negativ, so nimmt es den Wert -1 an (bei 0 wird SGN ebenfalls 0). Die Funktion dient also zur Bestimmung des Vorzeichens.

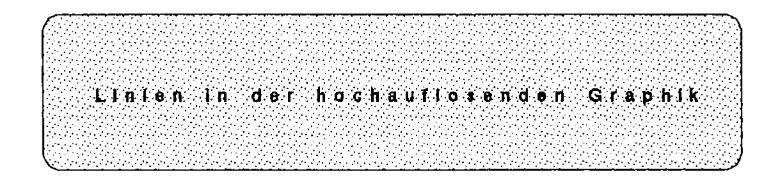
In Zeile 11060 wird jeder Punkt, der angesteuert wird dupliziert, so daß ein doppelt breiter Punkt entsteht. Dies ist notwendig, da einzelne Punkte, die in x-Richtung keinen Nachbarn besitzen entweder gar nicht oder nur sehr schwach zu sehen sind. So, und jetzt viel Spaß bei Ihrer Linienkreation.

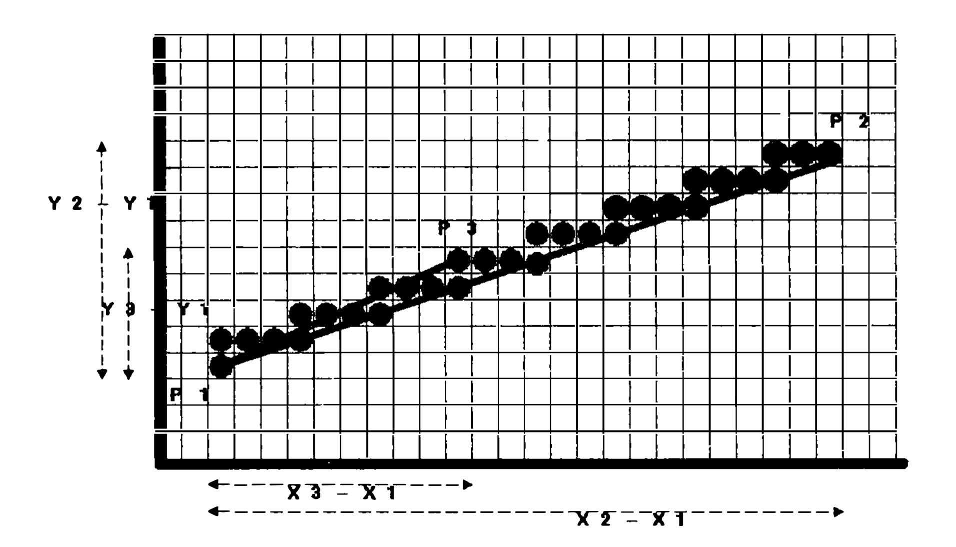
Linienberechnung in Assembler:

Wenn Sie das vorstehende Programm des öfteren für Ihre Linien und Zeichnungen verwenden wollen, so wird Ihnen die lange Wartezeit, die zum Zeichnen benötigt wird, recht bald unbequem. In diesem Fall sollte uns ein Blick hinter die Kulissen einer in Maschinensprache (Assembler) geschriebenen Routine nicht zu schade sein. Sie werden staunen, welchen Geschwindigkeitsvorteil eine solche Assemblerroutine bringen wird (s. Supergraphik).

Gehen wir die Sache einmal ganz ruhig an: Natürlich werden wir hier versuchen, einen Algorithmus, also eine Rechenvorschrift zu finden, in der nicht so viel zeitintensiv herumgerechnet werden muß. Dazu sehen wir uns einmal an, wie eine sogenannte Linie in Wirklichkeit auf dem Bildschirm ausschaut (s. Abbildung). Was uns da als schöne, gleichmäßige Linie erscheint, nimmt in Wahrheit einen recht wilden Zickzackverlauf. Wir werden nun nicht die Koordinaten jedes Punktes einzeln errechnen, sondern lediglich feststellen, in welche Richtung

(hoch, runter, rechts oder links) der nächste Punkt gezeichnet werden soll. Dadurch muß jedesmal nur eine einfache Addition durchgeführt werden.





Wie wir oben gesehen haben, können wir die Steigung einer Geraden durch die Formel m=(y2-yl)/(x2-xl) berechnen. Nehmen wir an, wir befinden uns während des Zeichnens unserer Linie gemäß obiger Abbildung im Punkt p3. Wie deutlich zu sehen ist, befindet sich dieser Punkt nicht auf, sondern vielmehr "über" unserer Ideallinie. Die Steigung der Geraden pl nach p3 ist also größer, als die Idealgerade pl nach p2. Aus diesem Grunde müssen wir den nächsten Punkt rechts neben dem zuletzt gezeichneten Punkt setzen, da sich nur so im nächsten Schritt die Steigung von pl->p3 wieder an die der Gerade pl->p2 annähert.

Wir können nun also unser Gesetz formulieren:
Ist die Steigung der Geraden zwischen Start- und aktuellem Punkt größer als die der Idealgeraden, so zeichnen wir den nächsten Punkt rechts neben den vorherigen, ist sie dagegen kleiner oder gleich, so wird der nächste Punkt darüber gezeichnet. Dies gilt natürlich nur dann, wenn man grundsätzlich von unten nach oben eine Linie mit positiver Steigung zeichnet. Doch realisieren wir erst einmal diesen einen Fall.

Im folgenden sei ma die Steigung der Gerade zwischen Start- und aktuellem Punkt (p1->p3) und mi die Steigung der Ideallinie (p1->p2). Dm ist dann die Differenz dieser Werte:

Dm = mi-ma

Folgende zwei Fälle sind zu unterscheiden:

a) Dm > 0 => Schritt nach oben
b) Dm < 0 => Schritt nach rechts

Für die Steigungen setzen wir nun die entsprechenden Punkt-Steigungs-Formeln (s.o.) ein:

```
ma = (y3-y1)/(x3-x1)
mi = (y2-y1)/(x2-x1)
```

Dann ergibt sich:

```
Dm                  = (y2-y1)/(x2-x1) - (y3-y1)/(x3-x1)    <=>
Dm*(x3-x1)*(x2-x1) = (y2-y1)*(x3-x1) - (y3-y1)*(x2-x1)
```

Da x3 und x2 immer größer sind als x1, ist auch der Ausdruck (x3-x1)*(x2-x1) immer größer null, ändert also nichts an dem Vorzeichen der linken Seite der Gleichung. Da aber nun interessant ist, ob Dm größer oder kleiner Null ist, kann man die beiden Klammern auf der linken Gleichungsseite getrost wegfallen lassen und wir erhalten wieder:

Dm = (y2-y1)*(x3-x1) - (y3-y1)*(x2-x1)

Streng mathematisch ist diese Gleichung natürlich falsch. Gleich sind natürlich nur noch die Vorzeichen der beiden Gleichungsseiten. Für unsere Zwecke geht das aber in Ordnung.

Bewegen wir uns nun nach rechts, so erhöht sich x3, also (x3-x1), entsprechend um 1, bewegen wir uns dagegen nach oben, so erhöht sich y3, also (y3-y1), um 1. Das neue Dm (Kurz NDm) sieht für den ersten Fall dann so aus:

```
NDm = (y2-y1)*((x3-x1)+1) - (y3-y1)*(x2-x1)          <=>
NDm = (y2-y1)*(x3-x1) + (x3-x1) - (y3-y1)*(x2-x1)    <=>
NDm = Dm + (x3-x1)
```

Entsprechend für den zweiten Fall:

```
NDm = (y2-y1)*(x3-x1) - ((y3-y1)+1)*(x2-x1)          <=>
NDm = (y2-y1)*(x3-x1) - (y3-y1)*(x2-x1) - (y3-y1)    <=>
NDm = Dm - (y3-y1)
```

Damit kann jedes NDm durch eine einfache Addition aus dem alten Dm errechnet werden. Dieser neue Wert wird dann auf sein Vorzeichen getestet, um wiederum zu entscheiden, in welche Richtung nun gezeichnet werden soll, die Koordinaten werden erhöht, das neue Dm wird errechnet usw.

Bisher haben wir diese Ableitung nur für Geraden mit positiver Steigung dargestellt. Bei Geraden mit m<0 (negative Steigung), also y3 kleiner y1, gelten die gleichen Regeln und Formeln, wenn man die Vorzeichen von (y2-y1) und (x2-x1) umkehrt, also den Betrag dieser Werte berechnet. Gleichzeitig muß natürlich statt nach rechts nach links und statt nach oben nach unten gezeichnet werden.

Dieses ganze Verfahren wird solange wiederholt, bis x3=x2 und y3=y2. Um nicht dauernd diese Differenz zu berechnen, wird einfach die Anzahl der zu zeichnenden Punkte anz gezählt:

```
anz= abs(x2-x1) + abs(y2-y1)
```

abs ist die Absolutfunktion, d.h. Vorzeichen werden weggestrichen. Ist die Anzahl der zu zeichnenden Punkte erreicht, ist die Linie fertig.

Im Supergraphik-Listing finden Sie die entsprechenden Routinen unter HLINE, die Vektorroutinen unter OBEN, UNTEN, RECHTS, LINKS.

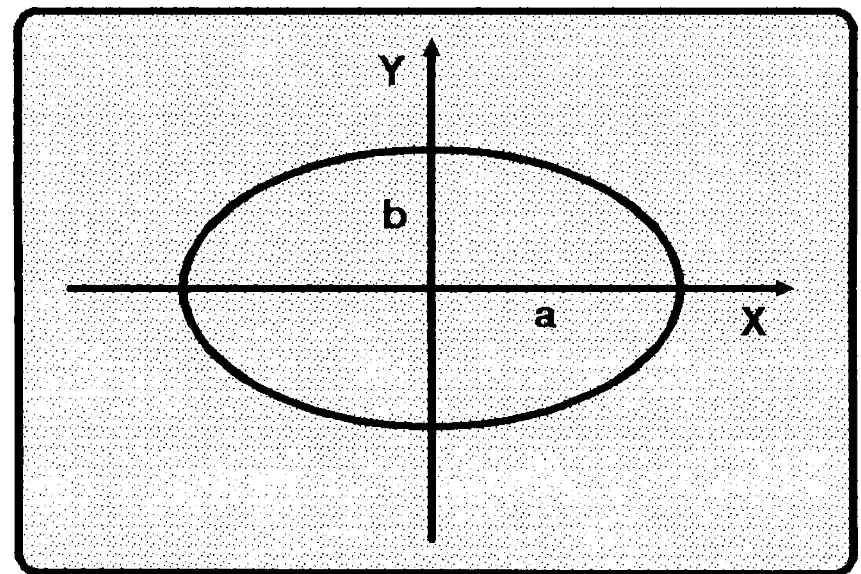
5.2.2.3 Ellipse/Kreis

Eine weitere wichtige und viel verwendete Figur ist der Kreis oder allgemeiner die Ellipse. Mit Ihnen lassen sich schöne Effekte erzeugen. Auch hierzu wird Ihnen im folgenden eine kleine Demonstrationsroutine angegeben, mit der Sie Ellipsen bzw. Kreise zeichnen können. Doch vorher sollten wir für die Interessierten unter Ihnen die mathematischen Grundlagen darlegen, die für das Verständnis dieser Funktion vonnöten sind. Wir werden uns in diesem Buch mit insgesamt drei Möglichkeiten der Kreis- bzw. Ellipsenerzeugung beschäftigen. Die erste etwas einfacher zu verstehende wird hier angeführt. Im Anschluß daran erfahren Sie etwas über die sogeannte Achtelkreistechnik. Die dritte Möglichkeit, sie resultiert aus der Verwendung sogenannter Polarkoordinaten und erlaubt das Zeichnen von Kreisbögen, fanden Sie bereits unter dem Abschnitt "Kuchendiagramme" im 3. Kapitel (Kap. 3.1.3). Doch hier seien Sie zunächst in die übliche Darstellungsweise eingeführt:

In unserer Routine gehen wir von der sogenannten Mittelpunktsgleichung der Ellipse aus:

```
x^2      y^2
---  +   ---   =  1
 a        b
```

Dabei bedeuten x und y die jeweiligen Koordinaten der Randpunkte der Ellipse. a ist der Radius in x-Richtung und b derjenige in y-Richtung. Der Mittelpunkt der Ellipse liegt im Koordinatenursprung (x=0/y=0):



Um diese Gleichung in unser Programm einzufügen, müssen wir sie zunächst einmal nach y auflösen:

$$y = b * \sqrt{1 - \frac{x^2}{a^2}}$$

Mit dieser recht kompliziert aussehenden Gleichung können wir nun die Punkte eines Ellipsenrandes berechnen. Dabei ist jedoch zu beachten, daß dabei stets nur gleichzeitig ein Bogen von 90 Grad gezeichnet werden kann, da eine Ellipse streng genommen keine Funktion darstellt (Relation). Um die vier anderen Bögen zu zeichnen, müssen wir die Vorzeichen von x und y umkehren. Für x geschieht das im unten stehenden Programm durch die FOR...NEXT-Schleife in Zeilen 11150 - 11190, in dem F2 nacheinander die Werte -1 und 1 annimmt. y dagegen wird in Zeile 11180 negiert. Da die obige Gleichung nur für Ellipsen mit dem Mittelpunkt bei x=0 und y=0 gilt, müssen wir entsprechende Summanden zu x und y hinzufügen, wie unten gezeigt.

Wollen Sie mit unten stehendem Programm einen Kreis zeichnen, so müssen Sie a und b (also die Speicher XR/YR) und damit die beiden Radien gleich groß werden lassen, weil ein Kreis lediglich einen Sonderfall einer Ellipse darstellt.

```
100 REM        ****************
110 REM        **            **
120 REM        **  ELLIPSE   **
130 REM        **            **
140 REM        ****************
150 REM
160 V=53248 : SA=8192
170 GOSUB 10000 : REM GRAPHIK EIN
180 FA = 1*16 + 0 : GOSUB 10400 : REM FARBE SETZEN
190 GOSUB 10200 : REM GRAPHIK LOESCHEN
270 XR=40:YR=20:XM=160:YM=100 :REM X/Y-RADIUS===MITTELPUNKTKOORDIN ATEN
280 GOSUB 11100 : REM ELLIPSE
290 WAIT 198,255 : REM AUF TASTE WARTEN
300 GOSUB 10600 : REM GRAPHIK AUS
310 END
320 REM
10000 REM ****************************
10020 REM **  GRAPHIK EINSCHALTEN  **
10040 REM ****************************
10050 REM
```

```
10070 POKE V+17, PEEK(V+17) OR (8+3)*16 : REM GRAPHIK EIN
10080 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10090 POKE V+24, PEEK(V+24) OR 8 : REM GRAPHIK NACH $2000 (8192)
10100 RETURN
10110 REM
10200 REM  *************************
10220 REM  **  GRAPHIK LOESCHEN  **
10240 REM  *************************
10250 REM
10270 FOR X=SA TO SA+8000 : POKE X,0 : NEXT X
10300 RETURN
10310 REM
10400 REM   **********************
10420 REM   **  FARBE LOESCHEN  **
10440 REM   **********************
10450 REM
10460 BF = 1024 : REM BASISADRESSE DES VIDEORAM
10480 FOR X=BF TO BF+1000 : POKE X,FA : NEXT X
10510 RETURN
10520 REM
10600 REM ***************************
10620 REM **  GRAPHIK AUSSCHALTEN  **
10640 REM ***************************
10650 REM
10670 POKE V+17, PEEK(V+17) AND 255-6*16 : REM GRAPHIK AUS
10680 POKE V+22, PEEK(V+22) AND 255-16 : REM MULTICOLOR AUS
10690 POKE V+24, PEEK(V+24) AND 255-8 : REM ZEICHENSATZ WIEDER NAC H
$1000 (4096)
10695 RETURN
10700 REM ***********************
10720 REM **  PUNKTBERECHNUNG  **
10730 REM **     (SETZEN)      **
10740 REM ***********************
10750 REM
10760 RA = 320 * INT(YK/8) + (YK AND 7)
10770 BA = 8 * INT(XK/8)
10780 MA = 2^(7-(XK AND 7))
10790 AD = SA + RA + BA
10800 POKE AD, PEEK(AD) OR MA
10810 RETURN
11100 REM
11110 REM ***********************
11120 REM **  ELLIPSE ZEICHNEN  **
11130 REM ***********************
11140 REM
11150 FOR F2=-1 TO 1 STEP 2 : REM RECHTS/LINKS-FLAG
11160 FOR X=0 TO F2*(XR) STEP F2
11170 ZK = YR * SQR(1-X^2/XR^2):XK=X+XM : REM KREISGLEICHUNG
11180 YK = YM + ZK:GOSUB 10760:YK = YM - ZK:GOSUB 10760 : REM PUNK TE
OBEN/UNTEN
11190 NEXT X,F2:RETURN
```

Die 4 Übergabeparameter sind:

```
XM/YM : Koordinaten des Mittelpunktes
XR/YR : x-/y-Radius (a und b)
```

Die Achtelkreismethode:

Es gibt nun noch eine zweite, sehr schnelle Methode, einen Kreis zu zeichnen, die sogenannte Achtelkreismethode. Hierbei werden nach jeder Berechnung insgesamt 8 verschiedene Punkte des Kreises gezeichnet. So reduziert sich die Zeit zum Zeichnen des Kreises auf etwa ein achtel. Ausgehend von den 4 Vierteln (0, 90, 180 und 270 Grad) wird der Kreis so jeweils in und gegen den Uhrzeigersinn gezeichnet. Dabei geht man folgendermaßen vor:

Angenommen, man hat die Koordinaten eines Kreispunktes bei 1 Grad nach dem alten Algorithmus (s.o.) berechnet und den entsprechenden Punkt des Kreises gezeichnet. Dieser befindet sich nun im rechten oberen Viertel des Kreises. Da ein Kreis nun ein sehr symmetrisches Objekt ist, kann dieser Punkt nun an dem senkrechten Durchmesser des Kreises gespiegelt werden. Die y-Koordinate dieses dadurch entstehenden neuen Punktes (im linken oberen Viertel) ist identisch mit der des Ursprungpunktes. Lediglich die x-Koordinate ist nach folgender Formel umzurechnen:

```
x2 = xmitte-(x1-xmitte)
x2 = 2*xmitte-x1
```

Dabei ist xmitte die x-Koordinate des Kreismittelpunktes. Da man zu den Koordinaten ja sowieso erst zum Schluß kurz vor dem Zeichnen die Mittelpunktkoordinaten hinzuaddiert (s.o.), braucht man in der obigen Gleichung die Rechnerei mit xmitte gar nicht, sondern rechnet einfach:

```
x2U = -x1U
```

Dabei sind x2U und x1U die Koordinaten des Kreises mit dem Mittelpunkt im Ursprung.

Damit kommt man sehr "billig" an einen zweiten Kreispunkt. Doch nicht genug. Spiegelt man diese beiden Punkte um den waagerechten Durchmesser, so erhält man wieder zwei neue Punkte. Diese beiden Punkte errechnet man folgendermaßen:

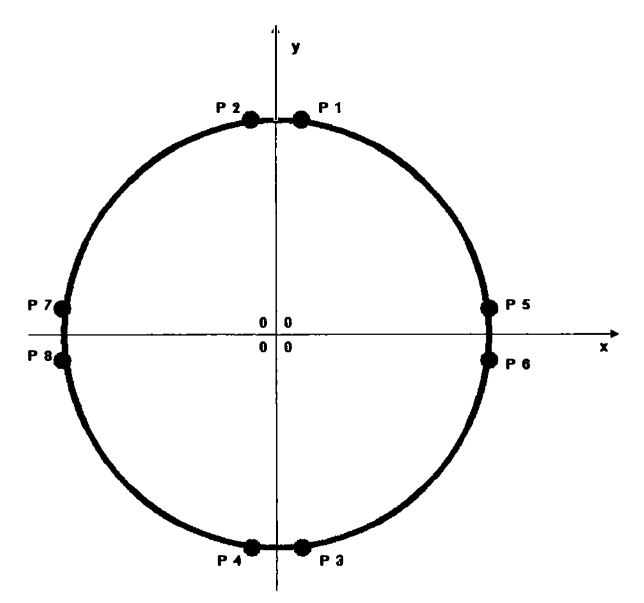
```
x3U =  x1U     und     x4U =  x2U = -x1U
y3U = -y1U     und     y4U = -y2U = -y1U
```

Auch hier sind (x3U,y3U) und (x4U,y4U) die Koordinaten der beiden in der unteren Hälfte des Kreises (um den Nullpunkt) liegenden Punkte.

Durch eine Drehung dieser nunmehr 4 Punkte um den Nullpunkt erhält man 4 weitere Punkte, die jeweils rechts und links vom Kreismittelpunkt liegen. Diese Punkte p5,p6,p7,p8 erhält man durch folgende Rechnungen:

```
x5U = y1U            und   y5U =  x1U
x6U = y1U            und   y6U = -x1U
x7U = y3U = -y1U     und   y7U =  x1U
x8U = y3U = -y1U     und   y8U = -x1U
```

Die untenstehende Zeichnung mag diesen Sachverhalt noch erläutern. Man kann also durch einfache Vorzeichenwechsel 8 verschiedene Koordinaten des Kreises berechnen. Nach diesen Spiegelungen und Drehungen addieren wir jeweils noch die Mittelpunktskoordinaten hinzu und schon liegen wir wieder richtig.



Der Nachteil dieser Kreisberechnung: Es werden stets nur volle Kreise gezeichnet. Einfache Bögen sind dagegen nur mit viel Aufwand zu realisieren. Weiterhin fällt die Möglichkeit flach, die Routine auch für Polygone zu verwenden. Bei Ellipsen ist der letzte Schritt (die Drehung) nicht ohne weiteres möglich, hier kann also nur die Viertelkreistechnik verwandt werden.

Ein kleines Programm soll das oben Gesagte kurz veranschaulichen:

```
100 REM **************************
110 REM **                      **
120 REM ** ACHTELKREISTECHNIK **
130 REM **                      **
140 REM **************************
150 REM
160 REM INITIALISIERUNG
170 GMODE 0,1:GCLEAR :REM GRAPHIK EIN UND CLR
180 R=50          :REM RADIUS
190 XM=160:YM=100 :REM MITTELPUNKTSKOORDINATEN
200 PI=3.1415926  :REM KONSTANTE PI
210 SW=PI/180     :REM SCHRITTWEITE = 1 GRAD
220 REM
230 REM HAUPTSCHLEIFE
240 FOR W=0 TO 45*PI/180 STEP SW
250   XU1=R*COS(W)  :REM X-KOORDINATE
260   YU1=R*SIN(W)G
270   X1=XU1+XM
280   Y1=YU1+YM     :REM KOORINATEN GEMAESS
290   XN=-XU1+XM    :REM MITTELPUNKT VERSCHOBEN
300   YN=-YU1+YM
310   XO=-XU1+YM
320   YO=-YU1+XM
330   XS=XU1+YM
340   YS=YU1+XM
350   PLOT ,X1,Y1   :REM 8 PUNKTE ZEICHNEN
360   PLOT ,YS,XS
370   PLOT ,YS,XO
380   PLOT ,X1,YN
390   PLOT ,XN,YN
400   PLOT ,YO,XO
410   PLOT ,YO,XS
420   PLOT ,XN,Y1
430 NEXT W
```

5.3 Spriteprogrammierung

Eine der wohl faszinierendsten Eigenschaften Ihres Rechners ist die Fähigkeit, insgesamt 8 sogenannte Sprites gleichzeitig auf den Bildschirm zu bringen. Wenn Sie den entsprechenden Abschnitt im vierten Kapitel (Kap. 4.5) gelesen haben, dann besitzen Sie schon einen kleinen Überblick über die Spriteorganisation und die hardwaremäßige Verwirklichung dieser Bildschirmobjekte. Hier nun lernen Sie, wie Sie mit ihnen umgehen und was Sie dabei zu beachten haben.

5.3.1 Erstellung von Sprites

Das erste Problem jeder Spriteprogrammierung ist die Erstellung eines solchen Objektes, denn dies ist natürlich die Voraussetzung für jede Manipulation. Doch schon dies ist ein recht schwieriges Unterfangen, da die Ablegung der Sprites im Speicher relativ kompliziert ist.

Wie sie aus Kap. 4.5 wissen, besitzt ein Sprite eine Auflösung von 24x21 Punkten (in Multicolor: 12x21). Jeder Punkt wird durch ein Bit (zwei Bit bei MC) im Speicher repräsentiert. Je 8 Punkte sind somit in einem Byte zusammengefaßt. Um eine Zeile zu bestimmen, sind damit 24/8 = 3 Bytes notwendig. Diese drei stehen im Speicher direkt hintereinander. Die nächsten drei Bytes definieren dann die zweite Zeile und so fort.

Um ein Sprite zu erstellen, legt man sich vorzugsweise eine Schablone an, die sie im Anhang finden und sich am besten abzeichnen und mehrmals kopieren sollten (oder Sie verwenden den unten stehenden Spriteeditor). In diese Schablone können Sie jeden einzelnen Punkt Ihres Sprites mit Bleistift als kleines Kreuz (bzw. in Multicolor als Ziffer, stellvertretend für die jeweilige Farbe) eintragen und erhalten so ein vollständiges und übersichtliches Bild des zukünftigen Raumschiffes, Vogels oder Buchstabens. Doch Vorsicht! Sie sollten darauf achten, daß Sie jeweils mindestens zwei Punkte nebeneinander zeichnen, da ein einzelner, ohne linken oder rechten Nachbarn nicht, oder nur sehr schwach auf dem Bildschirm erscheint. Dies gilt nicht für Multicolor, da hier ein Punkt sowieso schon doppelte Breite besitzt.

Im Anschluß daran ersetzen Sie jedes Kreuzchen durch eine 1, jedes freie Feld durch eine 0 oder, falls Sie ein Mulcicolorsprite entwerfen, durch die binäre Zahl, die sich aus der eingetragenen Ziffer ergibt. Nun fassen Sie jeweils 8 dieser Nullen und Einsen zu einem Byte zusammen und errechnen sich nach der Konversionstabelle im Anhang die entsprechende Dezimalzahl. Auf diese Weise erhalten Sie insgesamt

63 Zahlen von 0 bis 255, die den Inhalt der 63 Bytes einer Spritedefinition wiederspiegeln.

Von BASIC aus gibt es verschiedene Möglichkeiten, Spritedefinitionen abzulegen und wieder einzulesen. Die erste und wohl einfachste ist die Speicherung dieser 63 Daten in DATA-Zeilen. Natürlich können Sie sie platzsparend möglichst eng hintereinander packen, doch zweckmäßigerweise und aus Gründen der Übersichtlichkeit sollten Sie in etwa die folgende Form besitzen:

```
1000 DATA 000,000,000
1010 DATA 000,000,000
1020 DATA 002,000,064
1030 DATA 001,000,128
1040 DATA 000,129,000
1050 DATA 000,066,000
1060 DATA 000,060,000
1070 DATA 000,126,000
1080 DATA 000,195,000
1090 DATA 001,141,128
1100 DATA 003,044,192
1110 DATA 031,255,248
1120 DATA 062,153,124
1130 DATA 125,066,190
1140 DATA 255,255,255
1150 DATA 001,255,128
1160 DATA 001,189,128
1170 DATA 003,060,192
1180 DATA 015,000,240
1190 DATA 015,000,240
1200 DATA 000,000,000
```

Sie sehen zwar nicht sofort, daß es sich hierbei um das Fahrzeug eines Außerirdischen handelt, doch die 3x21-Bytestruktur wird doch recht deutlich. Jede DATA-Zeile enthält hier die Information für eine Spritezeile. Um jedoch diese Daten in den eigentlichen Speicher zu lesen (in die bekannten Blöcke), müssen wir noch eine kleine Routine hinzufügen, die etwa so aussehen könnte:

```
100 AD = 13*64 : REM ADRESSE BLOCK 13
110 FOR X=0 TO 62
120  READ DT : REM 63 DATEN LESEN
130  POKE AD+X, DT : REM IN BOLCK 13 POKEN
140 NEXT X
```

Dieser Zusatz liest nacheinander die 63 Daten ein und schreibt sie in den Speicher (zu den Blöcken s.u.). Diese Form der Spritespeicherung benötigt jedoch eine ganze Menge Speicherplatz. Eine weitere, platzsparendere Möglichkeit der Speicherung ist das Ablegen eines Sprites auf Diskette oder Kassette z.B. als Sequentielles File. Auf diese Weise können Sie an jeder beliebigen Stelle des Programms ein Sprite einlesen, das Sie auf Diskette gespeichert halten. So ist es ihnen beispielsweise möglich, ganze Datenbanken auf einer Diskette anzulegen, aus denen sich Ihr Programm die notwendigen Teile ausliest.

Am Anfang der Erzeugung eines solchen Definitionsfiles stehen dabei wieder unsere DATAs. Mit Hilfe des folgenden Programms können Sie nun die einzelnen Werte aus den bekannten DATA-Zeilen herauslesen und als Sequentielles File auf Diskette ablegen:

```
10 OPEN 1,8,2,"SPRITE,S,W" :REM FILE ZUM SCHREIBEN EROEFFNEN
20 FOR X=0 TO 62
30  READ DT : REM 63 DATEN LESEN
40  PRINT#1, CHR$(DT);     :REM AUF DISKETTE SCHREIBEN
50 NEXT X                  :REM (ASCII-FORMAT)
60 CLOSE 1                 :REM FILE SCHLIESSEN
```

Der Name des entstehenden Files ist "SPRITE". Um diese Daten wieder einzulesen und direkt in den entsprechenden Speicher zu POKEn, dürfte Ihnen diese Routine behilflich sein:

```
10 AD = 13*64              :REM ADRESSE BLOCK 13
20 OPEN 1,8,2,"SPRITE,S,R" :REM SEQ. FILE ZUM LESEN EROEFFNEN
30 FOR X=0 TO 62
40  GET#1, DT$             :REM DATEN LESEN (ASCII-FORMAT)
50  POKE AD+X, ASC(DT$+CHR$(0)) :REM UND POKEN
60 NEXT X
70 CLOSE 1                 :REM FILE SCHLIESSEN
```

selbstverständlich gibt es noch die Möglichkeit, ein Sprite direkt als Programmfile abzuspeichern und ebenso einzuladen.

Wie Sie sehen, ist die ganze Sache ziemlich kompliziert und macht Ungeübten einiges zu schaffen. Aus diesem Grunde wird Ihnen im folgenden ein Programm vorgestellt, das Ihnen die Arbeit der Spriteerstellung wesentlich erleichtert. Dieser Spriteeditor, der teilweise in BASIC und Maschinensprache geschrieben wurde, gibt Ihnen kom-

fortable Möglichkeiten in die Hand, ein hochauflösendes Sprite zu erstellen und schließlich in Ihr Programm einzubauen. Er erzeugt Programmfiles, die auf die gleiche Art und Weise gelesen werden können, wie in der letzten Routine demonstriert, wenn Sie die Zeile 20 dort durch die folgende Zeile ersetzen:

```
20 OPEN 1,8,2,"SPRITE,P,R" : REM PROGRAMMFILE ZUM LESEN OEFFNEN
```

Sicher ist es eine ganze Menge Arbeit, dieses Programm[1] abzutippen, deshalb sollten Sie es sich gleich von Diskette laden.

```
100 REM ********************
110 REM **                **
120 REM **  SPRITEFORMER  **
130 REM **                **
140 REM ********************
150 REM
160 REM INITIALISIERUNG:
170 REM ****************
180 GOSUB2730 :REM MASCHINENROUTINEN EINLESEN
190 POKE 53280,0:POKE 53281,0 :REM HINTERGRUND-/RAHMENFARBE
200 POKE650,255 :REM ALLE ZEICHEN REPEAT
210 POKE 45,0:POKE 46,80:RUN 220 :REM BASICENDE=$5000
220 REM
230 REM MASCHINENROUTINEN:
240 REM ******************
250 IN%=18432 :REM INITROUTINE
260 PU%=18632 :REM PUNKT EINZEICHNEN
270 NE%=18567 :REM KOORDINATENSYSTEM
280 LA%=18503 :REM ZEICHENSATZ LADEN
290 SP%=18531 :REM ZEICHENSATZ SPEICHERN
300 CA%=18758 :REM CATALOG
310 BE%=18712 :REM BEFEHLSIDENTIFIZIERUNG
320 IV%=18830 :REM INVERTIEREN
330 VR%=18844 :REM VERSCHIEBEN-RECHTS
340 VL%=18870 :REM VERSCHIEBEN-LINKS
350 VO%=18895 :REM VERSCHIEBEN-OBEN
360 VU%=18935 :REM VERSCHIEBEN-UNTEN
370 Q  =  704 :REM SPRITEBLOCK-ADRESSE
380 V  =53248 :REM VIDEOCONTROLLER
390 REM
400 REM CONTROLZEICHEN:
410 REM ***************
420 C0$=CHR$(147) :REM BILDSCHIRM LOESCHEN
430 C1$=CHR$( 19) :REM HOME
440 C2$=CHR$(183) :REM HOCHSTRICH
```

1 Dieses Programm funktioniert nur ohne die Supergraphik

```
450 C8$=CHR$( 99)+CHR$( 99)+CHR$( 99):C3$=CHR$(117)+C8$+CHR$(105) :REM O
BERER FENSTERR.1
460 C4$=CHR$(106)+C8$+CHR$(107) :REM UNTERER SPRITEFENSTERRAND 1
470 C5$=CHR$(117)+C8$+C8$+CHR$(105) :REM OBERER RAND 2
480 C8$=CHR$(106)+C8$+C8$+CHR$(107) :REM UNTERER RAND 2
490 C9$=CHR$( 98) :REM MITTELSTRICH (SENKR)
500 C6$=CHR$( 18) :REM RVS ON
510 C7$=CHR$(146) :REM RVS OFF
520 NA%=828 :REM FILENAMENLAENGE($C800)
530 GA%=186 :REM GERAETEADRESSE($BA)
540 TA%=821 :REM TASTE/BEFEHLSCODE
550 SG%=  1 :REM SPRITEGROESSE
560 YK%=822 :REM Y-KOORD
570 XK%=823 :REM X-KOORD
580 REM
590 REM FARBEN DEFINIEREN:
600 REM ******************
610 DATA 144,  5, 28,159,156, 30, 31,158
620 DATA 129,149,150,151,152,153,154,155
630 DIM C$(16):FOR Y=0 TO 15:READ X:C$(Y)=CHR$(X):NEXT Y
640 N=1:F(0)=0:F(1)=1:V$=" ":SYS IN% :REM FARBEN/INIT
650 REM
660 REM LOESCHROUTINE (FELDAUFBAU):
670 REM ****************************
680 SYS IN% : REM SPRITE LOESCHEN
690 PRINT C0$
700 PRINT C1$;SPC(13);C$(7);"SPRITE-CREATION"
710 PRINT SPC(12);C$(1);"(C) BY AXEL PLENGE"
720 PRINT C$(4);:FOR X=1 TO 40:PRINT C2$;:NEXT X
730 PRINTC$(7)"  7"C$(6)"6543210"C$(7)"7"C$(6)"6543210"C$(7)"7"C$(6)"654
3210";
740 SYS NE% : REM NETZ ZEICHNEN
750 GOSUB 1820:PRINT:PRINT :REM STATUSFELD ERSTELLEN
760 PRINT:PRINT:PRINTTAB(30);C3$
770 FOR X=1 TO 3:PRINTTAB(30);C9$;"   "C9$:NEXT X:PRINTTAB(30);C4$ :REM
TESTSPRITE1
780 PRINTTAB(27);"  ";C5$;"  ":FOR X=1 TO 5:PRINTTAB(27);"  ";C9$;"
 ";C9$:NEXT X
790 PRINTTAB(27);"  ";C8$;"  "; :REM TESTSPRITE2
800 POKE 53248+21,3:X=0:Y=0 :REM SPRITES AN/X-,Y-KOORDINATE=0
810 REM
820 REM EINGABESCHLEIFE:
830 REM ****************
840 A=X+2:B=Y+4:GOSUB 2450 :REM POSITIONIEREN
850 POKE XK%,X:POKE YK%,Y:F=0 :REM KOORDINATEN UEBERGEBEN
860 PRINT C$(7);C6$;" ";CHR$(157); :REM BLINKPHASE AN
870 FOR S=1 TO 50:GETA$:IF A$<>"" THEN 890
880 NEXT S:SYS PU%:FOR S=1 TO 50:GET A$:IF A$="" THEN NEXT S:GOTO 860
:REM AUSSCHALTEN
890 REM
900 REM BEFEHLSERKENNUNG:
```

```
910 REM ******************
920 SYS PU%:C=ASC(A$):POKE TA%,C:SYS BE%:S=PEEK(TA%) :REM BEF-UEBE
RGABE/RUECKMELDUNG
930 REM VERTEILUNG:
940 ON S GOTO 1050,1050,1070,1070,1090,1090
950 ON S-6 GOTO 1110,1110,1910,1910,1910,1910
960 ON S-12 GOTO 1910,1910,1910,1910,650,1360
970 ON S-18 GOTO 1450,1490,1570,1130,2150
980 ON S-23 GOTO 1200,1970,1240,810
990 REM
1000 REM BEFEHLSBEARBEITUNG:
1010 REM ********************
1020 REM
1030 REM CURSORBEWEGUNG:
1040 REM ****************
1050 X=X+1:IF X=24 THEN X=0:GOTO 1090
1060 GOTO 810 :REM RECHTS
1070 X=X-1:IF X<0 THEN X=23:GOTO 1110
1080 GOTO 810 :REM LINKS
1090 Y=Y+1:IF Y=21 THEN Y=0
1100 GOTO 810 :REM RUNTER
1110 Y=Y-1:IF Y<0 THEN Y=20
1120 GOTO 810 :REM HOCH
1130 REM
1140 REM BEENDEN:
1150 REM ********
1160 A=2:B=15:GOSUB 2450 :REM POSITIONIEREN
1170 PRINT C6$;C$(7);"BEENDEN?";C7$;C$(6):INPUT T$
1180 IF T$="J" OR T$="JA" THEN SYS 64738 :REM KALTSTART
1190 GOTO 690
1200 REM
1210 REM CATALOG:
1220 REM ********
1230 PRINT C0$:SYS CA%:GOSUB 2490:GOTO 690
1240 REM
1250 REM VERSCHIEBUNG:
1260 REM **************
1270 GOSUB 2530:GOSUB 2440:PRINT C$(1);"VERSCHIEBUNG" :REM MELDEFELD
1280 PRINT TAB(27)"NACH:":PRINT TAB(27)"RECHTS(R),":PRINT TAB(27)"LINKS(
L),"
1290 PRINT TAB(27)"OBEN  (O),":PRINT TAB(27)"UNTEN(U):"
1300 GOSUB 2490
1310 IF T$="R" THEN SYS VR%:GOTO1350
1320 IF T$="L" THEN SYS VL%:GOTO1350
1330 IF T$="O" THEN SYS VO%:GOTO1350
1340 IF T$="U" THEN SYS VU%
1350 GOSUB 2530:GOTO 700
1360 REM
1370 REM SPRITEGROESSE:
1380 REM ***************
1390 ON SG%+1 GOSUB 1410,1420,1430,1440
```

```
1400 POKE V+23,A:POKE V+29,B:SG%=(SG%+1) AND 3:GOTO 810
1410 A=2:B=2:RETURN
1420 A=0:B=2:RETURN
1430 A=2:B=0:RETURN
1440 A=0:B=0:RETURN
1450 REM
1460 REM SPRITE INVERTIEREN:
1470 REM ********************
1480 SYS IV% : GOTO 700
1490 REM
1500 REM SPRITE SPEICHERN:
1510 REM ******************
1520 GOSUB 2530
1530  GOSUB  2440:PRINT  C6$;C$(1);"SPRITEAB-":PRINT  TAB(27);C6$;"SPE
ICHERUNG";C7$
1540 GOSUB 1700:IF F=1 THEN F=0:GOTO 1490 :REM EINGABE/FEHLERABFR.
1550 IF F=2 THEN F=0:GOTO 1630 :REM FEHLER
1560 SYS SP%:GOTO 1650 :REM SPEICHERN
1570 REM
1580 REM SPRITE LADEN:
1590 REM **************
1600 GOSUB 2530
1610 GOSUB 2440:PRINT C6$;C$(1);"SPRITE":PRINT TAB(27);C6$;"LADEN:";C7$
1620 GOSUB 1700:IF F=1 THEN F=0:GOTO 1570
1630 IF F=2 THEN F=0:GOTO 690
1640 SYS LA%
1650 REM FEHLERABFRAGE (NUR FUER DISK!):
1660 OPEN 1,8,15:INPUT#1,DS,DS$,DT,DB:CLOSE1
1670 IF DS<20 THEN 690 :REM OK
1680 PRINT:T$=STR$(DS)+","+DS$+","+STR$(DT)+","+STR$(DB)
1690 GOSUB 2600:PRINT T$:FOR S=1 TO 2000:NEXT S:GOTO 690 :REM BLINKEN
1700 REM
1710 REM NAMENEINGABE:
1720 REM **************
1730 A$="":PRINT:PRINT TAB(27)"FILENAME"C$(6):PRINT TAB(27);:INPUT A$:T=
LEN(A$)
1740 S=VAL(RIGHT$(A$,1))
1750  IF  S<>0  AND  LEFT$(RIGHT$(A$,2),1)=";"  THEN  T=T-2:POKE  GA,S  :R EM
GERAETEADR.
1760 IF T=0 THEN F=2:RETURN :REM KEIN NAME
1770 IF T>17 THEN 1800
1780 REM NAMEN AN MASCHINENROUTINEN:
1790 POKE NA%,T:FOR S=1 TO T:POKE NA%+S,ASC(MID$(A$,S,1)):NEXT S:RETURN
1800 PRINT CHR$(145);:T$=C6$+"LAENGE!"+C7$:GOSUB 2590 :REM FEHLERMELDUNG
1810 PRINT C$(6):F=1:RETURN
1820 REM
1830 REM STATUSFELD ERSTELLEN:
1840 REM **********************
1850 A=27:B=4:GOSUB 2450
1860 GOSUB 2530
1870 A=27:B=5:GOSUB 2450
```

```
1880 PRINT TAB(27);C$(7);"FARBEN:"
1890 PRINT TAB(27);C$(2);:FOR S=1 TO 7:PRINT CHR$(163);:NEXT S:PRINT C$(
6)
1900 FOR S=0 TO 1:PRINT TAB(27);"GRDF.";S;":";F(S):NEXT S:RETURN
1910 REM
1920 REM PLOT:
1930 REM *****
1940 S=((S-12) AND 2)/2 :REM PLOTFARBE FESTSTELLEN
1950 T=X/8:AD=INT(T):T=2^(7-8*(T-AD)):AD=Y*3+AD+Q
1960 POKE AD,PEEK(AD) AND (255-T) OR S*T:GOTO 810
1970 REM
1980 REM FARBENWAHL:
1990 REM ***********
2000 PRINTCHR$(147)
2010 A=0:B=4:GOSUB 2450:PRINT TAB(4);C$(1)"F"C$(2)"A"C$(3)"R"C$(4)"B"C$(
5)"E";
2020 PRINT C$(6)"N"C$(7)"W"C$(4)"A"C$(6)"H"C$(2)"L"C$(7)":"
2030 PRINT TAB(4);C$(1);CHR$(172);:FOR S=1 TO 32:PRINT CHR$(162);:NEXT S
:PRINT CHR$(187)
2040 FOR S=1 TO 2:PRINT TAB(4);C6$;CHR$(161);
2050 FOR T=0 TO 15:PRINT C$(T);"  ";:NEXT T:PRINT C$(1);C7$;CHR$(161):NE
XT S
2060 PRINT TAB(4);C6$;CHR$(161);" 0 1 2 3 4 5 6 7 8 9101112131415";C7$;C
HR$(161)
2070 PRINT:PRINT C$(6);"    FUER GRUNDFARBENNR.(F1/F3): ";
2080 GOSUB 2490:T=ASC(T$)-133 :REM FUNKTIONSTASTE
2090 IF T<0 OR T>1 THEN GOSUB 2590:GOTO 690 :REM FEHLER
2100 IF T>1 THEN T=T-4
2110 PRINT T:T$="":INPUT "    FARBE ";T$:S=ABS(INT(VAL(T$)))
2120 IF T$="" OR S>15 THEN GOSUB 2590:GOTO690 :REM FEHLER
2130 F(T)=S:POKE V+33,F(0) :REM HINTERGRUNDFARBE SETZEN
2140 POKE V+39,F(1):POKE V+40,F(1):GOTO 690 :REM SPRITEFARBE SETZEN
2150 REM
2160 REM BEFEHLSSATZ:
2170 REM ************
2180 POKE V+21,0 : REM SPRITES AUS
2190 PRINT C0$;C6$;C$(2)"              ";C$(7);
2200 PRINT "BEFEHLSSATZ";C$(2);"              ";C7$;
2210 PRINT C$(4);:FOR S=1 TO 40:PRINT CHR$(184);:NEXT S:PRINT
2220 PRINT C$(1)" NR.  "C6$"BEFEHL "C7$"-"C$(5)" FUNKTION"C$(4)
2230 FOR S=1 TO 10:PRINT "----";:NEXT S
2240 PRINT C$(1)" (1)  "C6$"(><...)"C7$"-"C$(5)" CURSORBEWEGUNGEN"
2250 PRINT C$(1)"      "C6$"(2QWA )"C7$;C$(5)
2260 PRINT C$(1)" (2)  "C6$"(F1-F8)"C7$"-"C$(5)" PLOT IN FARBEN 0-15"
2270 PRINT C$(1)" (3)  "C6$"(F)    "C7$"-"C$(5)" FARBEN 0-15 F. F1 /3
DEF."
2280 PRINT C$(1)" (4)  "C6$"(B)    "C7$"-"C$(5)" BEFEHLSSATZ"
2290 PRINT C$(1)" (5)  "C6$"(G)    "C7$"-"C$(5)" SPRITE GROESSE"
2300 PRINT C$(1)" (6)  "C6$"(I)    "C7$"-"C$(5)" SPRITE INVERTIERE N"
2310 PRINT C$(1)" (7)  "C6$"(V)    "C7$"-"C$(5)" SPRITE VERSCHIEBE N"
2320 PRINT C$(1)" (8)  "C6$"(L)    "C7$"-"C$(5)" SPRITE LOESCHEN"
```

```
2330 PRINT C$(1)" (9)  "C6$"(CTRLG)"C7$"-"C$(5)" GET-SPRITE LADEN"
2340 PRINT C$(1)"(10)  "C6$"(CTRLS)"C7$"-"C$(5)" SAVE-SPRITE SPEIC HERN"
2350 PRINT C$(1)"(11)  "C6$"(C)    "C7$"-"C$(5)" DIREKTORY/CATALOG "
2360 PRINT C$(1)"(12)  "C6$"(CTRLX)"C7$"-"C$(5)" BEENDEN"
2370 GOSUB 2490:POKE V+21,3:GOTO 690 :REM WARTEN+SPRITES AN
2380 REM
2390 REM UNTERPROGRAMME:
2400 REM ****************
2410 REM
2420 REM POSITIONIERUNG:
2430 REM ****************
2440 A=27:B=5 :REM MELDEFELD
2450 PRINT C1$;:FOR S=2 TO B:PRINT:NEXT S:PRINT TAB(A);:RETURN
2460 REM
2470 REM TASTENEINGABE:
2480 REM ***************
2490 WAIT 198,255:GET T$:RETURN
2500 REM
2510 REM MELDEFELD LOESCHEN:
2520 REM ********************
2530 GOSUB 2440
2540 FORS=1TO6:PRINTTAB(27);:FORT=1TO4:PRINT"   ";:NEXT T: PRINT:NEXT S
 :REM MELDEFELD LOESCHEN
2550 RETURN
2560 REM
2570 REM FEHLERBLINKEN:
2580 REM ***************
2590 T$="UNZULAESSIG!"
2600 A=4:B=18:GOSUB2450:PRINTC$(1):FOR S=1 TO 9:PRINT TAB(4)T$:GOSUB 263
0:PRINT CHR$(145);
2610 PRINT TAB(4)"            ";
2620 PRINT CHR$(145):GOSUB 2630:NEXT S:PRINT TAB(4)"            ":F=1:RE
TURN
2630 FOR T=1 TO 75:NEXT T:RETURN :REM WARTESCHLEIFE
2640 REM
2650 REM **************************
2660 REM **                      **
2670 REM **  MASCHINENROUTINEN   **
2680 REM **                      **
2690 REM **************************
2700 REM
2710 REM DATAS WERDEN NACH DEM STARTEN GELOESCHT !!!
2720 REM
2730 FOR I = 1 TO 16 : READ X : NEXT I : REM VORDERE DATAS UEBERSPRINGEN
 (FARBEN)
2740 FOR I = 18432 TO 18969
2750 READ X : POKE I,X : S=S+X : NEXT
2760 DATA 162, 62,169,  0,157,192,  2,202, 16,250,169, 11
2770 DATA 141,248,  7,141,249,  7,169, 16,141,  0,208,169
2780 DATA 163,141,  1,208,169,  7,141,  2,208,169,201,141
2790 DATA   3,208,169,  3,141, 16,208,141, 21,208,169,  2
```

```
2800 DATA 141, 23,208,141, 29,208,169,  0,141, 27,208,141
2810 DATA  28,208,169,  1,141, 39,208,141, 40,208, 96,173
2820 DATA  60,  3,162, 61,160,  3, 32,249,253,169,  2,166
2830 DATA 186,160,  0, 32,  0,254,169,  0,162,192,160,  2
2840 DATA  76,213,255,173, 60,  3,162, 61,160,  3, 32,249
2850 DATA 253,169,  2,166,186,160,  0, 32,  0,254,169,192
2860 DATA 133,  2,169,  2,133,  3,169,  2,162,255,160,  2
2870 DATA  76,216,255,160,  0,169, 13, 32,210,255,152, 72
2880 DATA  56,233, 10,144, 12,168,233, 10,144,  4,168,169
2890 DATA  50, 44,169, 49, 44,169, 32, 32,210,255,152,  9
2900 DATA  48, 32,210,255,104,168,162,  0, 32,206, 72,169
2910 DATA  29, 32,210,255,232,224, 24,208,243,169,165, 32
2920 DATA 210,255,200,192, 21,208,194, 96,174, 55,  3,172
2930 DATA  54,  3,138, 72,152, 72,133,  2, 10,101,  2,133
2940 DATA   2,138,160,255, 56,233,  8,200,176,251,105,  8
2950 DATA 170,152, 24,101,  2,168,169,  0, 56,106,202, 16
2960 DATA 252, 57,192,  2,208,  3,160,  0, 44,160,  6,162
2970 DATA   6,185, 12, 73, 32,210,255,200,202,208,246,104
2980 DATA 168,104,170, 96,146, 31,111, 31,146,157, 18, 28
2990 DATA  32, 31,146,157,173, 53,  3,162,  0,160, 28,232
3000 DATA 221, 43, 73,240,  3,136,208,247,142, 53,  3, 96
3010 DATA  87, 29, 81,157, 65, 17, 50,145,133,137,134,138
3020 DATA 135,139,136,140, 76, 71, 73, 19,  7, 24, 66, 67
3030 DATA  70, 86,169, 36,133,  2,169,  1,162,  2,160,  0
3040 DATA  32,249,253,169,  2,166,186,160,  0, 32,  0,254
3050 DATA 169,  0,162,  0,160, 64,134, 95,132, 96, 32,213
3060 DATA 255,165, 95,164, 96, 32, 55,165,173,  0,  3, 72
3070 DATA 173,  1,  3, 72,169, 61,141,  0,  3,169,227,141
3080 DATA   1,  3, 32,195,166,104,141,  1,  3,104,141,  0
3090 DATA   3, 96,162, 62,189,192,  2, 73,255,157,192,  2
3100 DATA 202, 16,245, 96,162,  0,160,  3, 24,126,192,  2
3110 DATA 232,136,208,249,169,  0,106, 29,189,  2,157,189
3120 DATA   2,224, 63,208,233, 96,162, 63,160,  3, 24, 62
3130 DATA 191,  2,202,136,208,249,169,  0, 42, 29,194,  2
3140 DATA 157,194,  2,138,208,234, 96,162, 62,134,  2,160
3150 DATA  21,132,  3,166,  2,189,132,  2, 72,189,192,  2
3160 DATA 168,104,157,192,  2,152,202,202,202,198,  3,208
3170 DATA 239,166,  2,202,134,  2,224, 59,208,221, 96,162
3180 DATA   2,134,  2,160, 21,132,  3,166,  2,189,252,  2
3190 DATA  72,189,192,  2,168,104,157,192,  2,152,232,232
3200 DATA 232,198,  3,208,239,198,  2, 16,226, 96
3210 IF S <> 61707 THEN PRINT "FEHLER IN DATAS !!" : END
3220 PRINT "OK" : RETURN
```

```
                0010         .LS
                0020         ;
                0030         ;MASCHINENROUTINEN:
                0040         ;******************
                0050         ;
                0060         ;
                0070         .OS
                0080         .BA $4800          ;STARTADRESSE
                0090         .MC $0800
                0100         ;
                0110         ;SPRUNGADRESSEN UND REGISTER:
                0120         ;****************************
                0130         ;
                0140 CONASS  .DE 1
                0150 CHROUT  .DE $FFD2          ;ZEICHENAUSGABE
                0160 FNPAR   .DE $FDF9          ;FILENAMENPARAMETER SETZEN
                0170 FPAR    .DE $FE00          ;FILEPARAMETER SETZEN
                0180 SAVE    .DE $FFD8          ;SPEICHERN AUF DISK/KASSETTE
                0190 LOAD    .DE $FFD5          ;LADEN VON DISK/KASSETTE
                0200 V       .DE $D000          ;VIDEOCONTROLLER (53248)
                0210 BLOCK   .DE 704            ;SPRITEBLOCK 11
                0220 BLOCKN  .DE 11             ;BLOCKNUMMER 11
                0230 LAENGE  .DE $033C          ;FILENAMENLAENGE
                0240 MODUS   .DE $0334          ;SPEICHERMODUS
                0250 TASTE   .DE $0335          ;BEFEHLSTASTENDRUCK
                0260 YKOORD  .DE $0336          ;FELD-Y-KOORDINATE
                0270 XKOORD  .DE $0337          ;FELD-X-KOORDINATE
                0280         ;
                0290         ;TESTSPRITE LOESCHEN+PARAMETER:
                0300         ;******************************
                0310         ;
4800- A2 3E     0320 INIT    LDX #62            ;63 BYTES
4802- A9 00     0330         LDA #$00
4804- 9D C0 02  0340 I1      STA BLOCK,X        ;BLOCK 11 LOESCHEN
4807- CA        0350         DEX
4808- 10 FA     0360         BPL I1
480A- A9 0B     0370         LDA #BLOCKN        ;BLOCK 11
480C- 8D F8 07  0380         STA $07F8          ;POINTER SPRITE 0 AUF 11
480F- 8D F9 07  0390         STA $07F9          ;POINTER SPRITE 1 AUF 11
4812- A9 10     0400         LDA #16
4814- 8D 00 D0  0410         STA V+0            ;SPRITE 0:X-KOORD. HIGHBYTE
4817- A9 A3     0420         LDA #163
4819- 8D 01 D0  0430         STA V+1            ;Y-KOORDINATE
481C- A9 07     0440         LDA #7
481E- 8D 02 D0  0450         STA V+2            ;SPRITE 1:X-KOORD. HIGHBYTE
4821- A9 C9     0460         LDA #201
4823- 8D 03 D0  0470         STA V+3            ;Y-KOORDINATE
4826- A9 03     0480         LDA #%00000011     ;X-K. HIGHBYTES=1
4828- 8D 10 D0  0490         STA V+16
482B- 8D 15 D0  0500         STA V+21           ;SPRITES EINSCHALTEN
482E- A9 02     0510         LDA #%00000010
4830- 8D 17 D0  0520         STA V+23           ;SPRITE 1: Y-VERGROESSERUNG
4833- 8D 1D D0  0530         STA V+29           ;SPRITE 1: X-VERGROESSERUNG
4836- A9 00     0540         LDA #$00
4838- 8D 1B D0  0550         STA V+27           ;SPRITE-PRIORITAET
483B- 8D 1C D0  0560         STA V+28           ;NORMALFARBENSPRITES
483E- A9 01     0570         LDA #$01
4840- 8D 27 D0  0580         STA V+39           ;SPRITE 0: WEISS
4843- 8D 28 D0  0590         STA V+40           ;SPRITE 1: WEISS
4846- 60        0600         RTS                ;ZURUECK
                0610         ;
                0620         ;SPRITE LADEN:
```

```
                  0630        ;*************
                  0640        ;
4847- AD 3C 03    0650 LADEN  LDA LAENGE        ;NAMENLAENGE
484A- A2 3C       0660        LDX #L,LAENGE     ;FILENAMENADRESSE LOW-
484C- A0 03       0670        LDY #H,LAENGE     ;HIGHBYTE
484E- 20 F9 FD    0680        JSR FNPAR
4851- A9 02       0690        LDA #$02          ;LOGISCHE FILENUMMER
4853- A6 BA       0700        LDX *$BA          ;GERAETEADESSE
4855- A0 00       0710        LDY #$00          ;SECUNDAERADRESSE
4857- 20 00 FE    0720        JSR FPAR
485A- A9 00       0730        LDA #$00          ;LOAD/VERIFY-FLAG
485C- A2 C0       0740        LDX #L,BLOCK      ;STARTADRESSE (LOWBYTE)
485E- A0 02       0750        LDY #H,BLOCK      ;STARTADRESSE (HIGHBYTE)
4860- 4C D5 FF    0760        JMP LOAD
                  0770        ;
                  0780        ;SPRITE SPEICHERN:
                  0790        ;*****************
                  0800        ;
4863- AD 3C 03    0810 SPEICH LDA LAENGE        ;FILENAMENLAENGE
4866- A2 3C       0820        LDX #L,LAENGE     ;FILENAMENADRESSE (LOW)
4868- A0 03       0830        LDY #H,LAENGE     ;FILENAMENADRESSE (HIGH)
486A- 20 F9 FD    0840        JSR FNPAR
486D- A9 02       0850        LDA #$02          ;LOGISCHE FILENUMMER
486F- A6 BA       0860        LDX *$BA          ;GERAETEADRESSE
4871- A0 00       0870        LDY #$00          ;SECUNDAERADRESSE
4873- 20 00 FE    0880        JSR FPAR
                  0950        .LS
4876- A9 C0       0960        LDA #L,BLOCK
4878- 85 02       0970        STA *$02          ;STARTADRESSE (LOWBYTE)
487A- A9 02       0980        LDA #H,BLOCK
487C- 85 03       0990        STA *$03          ;STARTADRESSE (HIGHBYTE)
487E- A9 02       1000        LDA #$02          ;NULLSEITENADRESSE DER STARTADRESS
4880- A2 FF       1010        LDX #L,BLOCK+$3F  ;ENDADRESSE (LOWBYTE)
4882- A0 02       1020        LDY #H,BLOCK+$3F  ;ENDADRESSE (HIGHBYTE)
4884- 4C D8 FF    1030        JMP SAVE          ;SPEICHERE SPRITE
                  1450        .LS
                  1460        ;
                  1470        ;ARBEITSFELD HERSTELLEN:
                  1480        ;***********************
                  1490        ;
4887- A0 00       1500 NETZ   LDY #$00          ;ZEILENZAEHLER = 0
4889- A9 0D       1510 N0     LDA #$0D          ;CARRIGE RETURN
488B- 20 D2 FF    1520        JSR CHROUT
488E- 98          1530        TYA
488F- 48          1540        PHA
4890- 38          1550        SEC
4891- E9 0A       1560        SBC #10
4893- 90 0C       1570        BCC N1            ;LEERZEICHEN BEI Y<10
4895- A8          1580        TAY
4896- E9 0A       1590        SBC #10
4898- 90 04       1600        BCC N2            ;Y<20
489A- A8          1610        TAY
489B- A9 32       1620        LDA #'2
489D- 2C          1630        .BY $2C           ;NAECHSTEN BEF. UEBERSPRINGEN
489E- A9 31       1640 N2     LDA #'1
48A0- 2C          1650        .BY $2C           ;NAECHSTEN BEF. UEBERSPRINGEN
48A1- A9 20       1660 N1     LDA #$20          ;" " LEERZEICHEN
48A3- 20 D2 FF    1670        JSR CHROUT        ;AUSGEBEN
48A6- 98          1680        TYA               ;REST
48A7- 09 30       1690        ORA #$30          ;ASCII HERSTELLEN
48A9- 20 D2 FF    1700        JSR CHROUT        ;AUSGEBEN
48AC- 68          1710        PLA
48AD- A8          1720        TAY
48AE- A2 00       1730        LDX #$00
48B0- 20 CE 48    1740 N3     JSR PUNKT         ;EINEN PUNKT ZEICHNEN
48B3- A9 1D       1750        LDA #$1D          ;CURSOR RECHTS
```

```
48B5- 20 D2 FF  1760        JSR CHROUT
48B8- E8        1770        INX              ;NAECHSTER PUNKT
48B9- E0 18     1780        CPX #24          ;24 PUNKTE PRO ZEILE
48BB- D0 F3     1790        BNE N3
48BD- A9 A5     1800        LDA #$A5         ;STRICH (CHR$(165))
48BF- 20 D2 FF  1810        JSR CHROUT
48C2- C8        1820        INY              ;NAECHSTE ZEILE
48C3- C0 15     1830        CPY #21          ;21 ZEILEN
48C5- D0 C2     1840        BNE N0
48C7- 60        1850        RTS
                1860        ;
                1870        ;EINE KOORDINATE ZEICHNEN:
                1880        ;*************************
                1890        ;
48C8- AE 37 03  1900 PUNKT2 LDX XKOORD       ;ANSTEUERUNG DURCH BASIC
48CB- AC 36 03  1910        LDY YKOORD       ;X/Y-KOORDINATE
48CE- 8A        1920 PUNKT  TXA
48CF- 48        1930        PHA
48D0- 98        1940        TYA
48D1- 48        1950        PHA              ;KOORDINATEN RETTEN
48D2- 85 02     1960        STA *$02
48D4- 0A        1970        ASL A
48D5- 65 02     1980        ADC *$02         ;Y-KOORDINATE*3
48D7- 85 02     1990        STA *$02         ;UND RETTEN
48D9- 8A        2000        TXA
48DA- A0 FF     2010        LDY #$FF         ;ZAEHLER=0
48DC- 38        2020        SEC
48DD- E9 08     2030 P3     SBC #$08
48DF- C8        2040        INY
48E0- B0 FB     2050        BCS P3           ;X-KOORDINATE/8
48E2- 69 08     2060        ADC #$08         ;REST
48E4- AA        2070        TAX
48E5- 98        2080        TYA
48E6- 18        2090        CLC
48E7- 65 02     2100        ADC *$02         ;Y*3+INT(X/8)
48E9- A8        2110        TAY
48EA- A9 00     2120        LDA #$00
48EC- 38        2130        SEC              ;EIN BIT SETZEN
48ED- 6A        2140 P0     ROR A            ;RICHTIGES BIT HERAUSSUCHEN
48EE- CA        2150        DEX              ;REST ERNIEDIRIGEN
48EF- 10 FC     2160        BPL P0
48F1- 39 C0 02  2170        AND BLOCK,Y      ;ANDERE BITS DES SPRITEBYTES LOESC
48F4- D0 03     2180        BNE P1           ;BIT (=PUNKT) GESETZT?
48F6- A0 00     2190        LDY #$00         ;NEIN
48F8- 2C        2200        .BY $2C          ;BIT-BEFEHL
48F9- A0 06     2210 P1     LDY #$06         ;BIT GESETZT!
48FB- A2 06     2220        LDX #$06
48FD- B9 0C 49  2230 P2     LDA PKTTAB,Y     ;ZEICHEN AUS PUNKTDARSTELLUNGSTABE
4900- 20 D2 FF  2240        JSR CHROUT
4903- C8        2250        INY
4904- CA        2260        DEX
4905- D0 F6     2270        BNE P2           ;NAECHSTES ZEICHEN
4907- 68        2280        PLA
4908- A8        2290        TAY
4909- 68        2300        PLA
490A- AA        2310        TAX              ;KOORDINATEN WIEDERHOLEN
490B- 60        2320        RTS
                2330        ;
                2340        ;ZEICHEN FUER EINE KOORDINATE:
                2350        ;*****************************
                2360        ;
490C- 92 1F 6F  2370 PKTTAB .BY 146 031 111 031 146 157
490F- 1F 92 9D
4912- 12 1C 20  2380        .BY 018 028 032 031 146 157
4915- 1F 92 9D
                2390        ;
```

```
                  2400        ;TASTE ALS BEFEHL IDENTIFIZIEREN:
                  2410        ;********************************
                  2420        ;
4918- AD 35 03    2430 BEFIDE LDA TASTE         ;TASTENCODE
491B- A2 00       2440        LDX #$00          ;BEFEHLSCODE
491D- A0 1C       2450        LDY #$1C          ;27-1 BEFEHLE (ZAEHLER)
491F- E8          2460 B1     INX               ;BEFEHLSCODE ERHOEHEN
4920- DD 2B 49    2470        CMP BEFTAB-1,X    ;TASTE MIT TABELLE VERGLEICHEN
4923- F0 03       2480        BEQ B2            ;GEFUNDEN
4925- 88          2490        DEY               ;NAECHSTER BEFEHL
4926- D0 F7       2500        BNE B1            ;NOCH NICHT FERTIG
4928- 8E 35 03    2510 B2     STX TASTE         ;BEFEHLSCODE ALS RUECKMELDUNG
492B- 60          2520        RTS               ;ZURUECK ZU BASIC
                  2530        ;
                  2540        ;BEFEHLSTASTENTABELLE:
                  2550        ;********************
                  2560        ;
                  2570        ;    W/CRSR-RE/Q/CRSR-LI/A/CRSR-UN
492C- 57 1D 51    2580 BEFTAB .BY 087 029  081  157  065  017
492F- 9D 41 11
                  2590        ;    2/CRSR-OB/ F1/ F2/ F3/ F4
4932- 32 91 85    2600        .BY 050 145   133 137 134 138
4935- 89 86 8A
                  2610        ;    F5/ F6/ F7/ F8/ L / G
4938- 87 8B 88    2620        .BY 135 139 136 140 076 071
493B- 8C 4C 47
                  2630        ;    I/CTRL.S/CTRL.G/CTRL.X/ B
493E- 49 13 07    2640        .BY 073 019   007     024  066
4941- 18 42
                  2650        ;    C / F / V
4943- 43 46 56    2660        .BY 067 070 086
                  2670        ;
                  2680        ;DISKCATALOG:
                  2690        ;************
                  2700        ;
4946- A9 24       2710 CATALG LDA #$24          ;"$" ALS FILENAME
4948- 85 02       2720        STA *$02
494A- A9 01       2730        LDA #$01
494C- A2 02       2740        LDX #$02
494E- A0 00       2750        LDY #$00          ;S.O.
4950- 20 F9 FD    2760        JSR FNPAR
4953- A9 02       2770        LDA #$02
4955- A6 BA       2780        LDX *$BA          ;GERAETEADRESSE
4957- A0 00       2790        LDY #$00
4959- 20 00 FE    2800        JSR FPAR
495C- A9 00       2810        LDA #$00          ;LOAD/VERIFY-FLAG
495E- A2 00       2820        LDX #$00
4960- A0 40       2830        LDY #$40          ;STARTADRESSE
4962- 86 5F       2840        STX *$5F
4964- 84 60       2850        STY *$60
4966- 20 D5 FF    2860        JSR LOAD          ;LADE CATALOG WIE BASICPROGRAMM
4969- A5 5F       2870        LDA *$5F          ;SIMULIERE:
496B- A4 60       2880        LDY *$60          ;BASIC-PROGRAMMSTARTADRESSE
496D- 20 37 A5    2890        JSR $A537         ;BASICZEILEN BINDEN
4970- AD 00 03    2900        LDA $0300
4973- 48          2910        PHA
4974- AD 01 03    2920        LDA $0301         ;ORIGINALEN WARMSTARTVEKTOR
4977- 48          2930        PHA               ; RETTEN
4978- A9 3D       2940        LDA #$3D
497A- 8D 00 03    2950        STA $0300
497D- A9 E3       2960        LDA #$E3
497F- 8D 01 03    2970        STA $0301         ;UND AUF RTS SETZEN
4982- 20 C3 A6    2980        JSR $A6C3         ;LISTBEFEHL AUSFUEHREN
4985- 68          2990        PLA
4986- 8D 01 03    3000        STA $0301
4989- 68          3010        PLA
```

```
498A- 8D 00 03  3020        STA $0300         ;ALTEN VEKTOR WIEDERHOLEN
498D- 60        3030        RTS               ;ZURUECK ZU BASIC
                3040        ;
                3050        ;SPRITE INVERTIEREN:
                3060        ;******************
                3070        ;
498E- A2 3E     3080 INVERS LDX #62           ;62+1 BYTES
4990- BD C0 02  3090 IN1    LDA BLOCK,X       ;BYTE HOLEN
4993- 49 FF     3100        EOR #$FF          ;INVERTIEREN
4995- 9D C0 02  3110        STA BLOCK,X       ;ZURUECKSCHREIBEN
4998- CA        3120        DEX
4999- 10 F5     3130        BPL IN1           ;NAECHSTES BYTE
499B- 60        3140        RTS
                3150        ;
                3160        ;SPRITE VERSCHIEBEN:
                3170        ;******************
                3180        ;
499C- A2 00     3190 RECHTS LDX #$00          ;BYTEZAEHLER ABSOLUT
499E- A0 03     3200 RE1    LDY #$03          ;BYTEZAEHLER IN ZEILE
49A0- 18        3210        CLC
49A1- 7E C0 02  3220 RE2    ROR BLOCK,X       ;VERSCHIEBEN
49A4- E8        3230        INX
49A5- 88        3240        DEY
49A6- D0 F9     3250        BNE RE2
49A8- A9 00     3260        LDA #$00
49AA- 6A        3270        ROR A             ;LETZTES BIT IN AKU
49AB- 1D BD 02  3280        ORA BLOCK-3,X     ;UND AN ANFANG SETZEN
49AE- 9D BD 02  3290        STA BLOCK-3,X
49B1- E0 3F     3300        CPX #63
49B3- D0 E9     3310        BNE RE1
49B5- 60        3320        RTS               ;FERTIG
                3330        ;
                3340        ;
49B6- A2 3F     3350 LINKS  LDX #63           ;BYTEZAEHLER ABSOLUT
49B8- A0 03     3360 LI1    LDY #$03          ;BYTEZAEHLER IN ZEILE
49BA- 18        3370        CLC
49BB- 3E BF 02  3380 LI2    ROL BLOCK-1,X     ;VERSCHIEBEN
49BE- CA        3390        DEX
49BF- 88        3400        DEY
49C0- D0 F9     3410        BNE LI2
49C2- A9 00     3420        LDA #$00
49C4- 2A        3430        ROL A             ;LETZTES BIT IN AKU
49C5- 1D C2 02  3440        ORA BLOCK+2,X     ;UND ANS ENDE SETZEN
49C8- 9D C2 02  3450        STA BLOCK+2,X
49CB- 8A        3460        TXA
49CC- D0 EA     3470        BNE LI1
49CE- 60        3480        RTS               ;FERTIG
                3490        ;
                3500        ;
49CF- A2 3E     3510 OBEN   LDX #62           ;BYTEZAEHLER ABSOLUT
49D1- 86 02     3520        STX *$02
49D3- A0 15     3530 OB1    LDY #21           ;ZEILENZAEHLER
49D5- 84 03     3540        STY *$03
49D7- A6 02     3550        LDX *$02
49D9- BD 84 02  3560        LDA BLOCK-60,X    ;ERSTES BYTE HOLEN
49DC- 48        3570 OB2    PHA               ;IN ZWISCHENSPEICHER 1
49DD- BD C0 02  3580        LDA BLOCK,X       ;BYTE HOLEN
49E0- A8        3590        TAY               ;IN ZWISCHENSPEICHER 2
49E1- 68        3600        PLA               ;ZISCHENSPEICHER 1 HOLEN
49E2- 9D C0 02  3610        STA BLOCK,X       ;IN BYTE ABLEGEN
49E5- 98        3620        TYA               ;ZWISCHENSPEICHER 2 HOLEN
49E6- CA        3630        DEX
49E7- CA        3640        DEX
49E8- CA        3650        DEX               ;DARUEBER LIEGENDES BYTE
49E9- C6 03     3660        DEC *$03          ;NAECHSTE ZEILE
49EB- D0 EF     3670        BNE OB2
```

```
49ED- A6 02     3680          LDX *$02
49EF- CA        3690          DEX
49F0- 86 02     3700          STX *$02        ;NAECHSTE SPALTE
49F2- E0 3B     3710          CPX #59
49F4- D0 DD     3720          BNE OB1
49F6- 60        3730          RTS
                3740          ;
                3750          ;
49F7- A2 02     3760 UNTEN    LDX #02         ;BYTEZAEHLER ABSOLUT
49F9- 86 02     3770          STX *$02
49FB- A0 15     3780 UN1      LDY #21         ;ZEILENZAEHLER
49FD- 84 03     3790          STY *$03
49FF- A6 02     3800          LDX *$02
4A01- BD FC 02  3810          LDA BLOCK+60,X  ;LETZTES BYTE HOLEN
4A04- 48        3820 UN2      PHA             ;IN ZWISCHENSPEICHER 1
4A05- BD C0 02  3830          LDA BLOCK,X     ;BYTE HOLEN
4A08- A8        3840          TAY             ;IN ZWISCHENSPEICHER 2
4A09- 68        3850          PLA             ;ZISCHENSPEICHER 1 HOLEN
4A0A- 9D C0 02  3860          STA BLOCK,X     ;IN BYTE ABLEGEN
4A0D- 98        3870          TYA             ;ZWISCHENSPEICHER 2 HOLEN
4A0E- E8        3880          INX
4A0F- E8        3890          INX
4A10- E8        3900          INX             ;DARUNTER LIEGENDES BYTE
4A11- C6 03     3910          DEC *$03        ;NAECHSTE ZEILE
4A13- D0 EF     3920          BNE UN2
4A15- C6 02     3930          DEC *$02        ;NAECHSTE SPALTE
4A17- 10 E2     3940          BPL UN1
4A19- 60        3950          RTS
                3960          .EN
END OF ASSEMBLY'


 ---  LABEL FILE:  --


B1 =491F              B2 =4928
BEFIDE =4918          BEFTAB =492C
BLOCK =02C0           BLOCKN =000B
CATALG =4946          CHROUT =FFD2
CONASS =0001          FNPAR =FDF9
FPAR =FE00            I1 =4804
IN1 =4990             INIT =4800
INVERS =498E          LADEN =4847
LAENGE =033C          LI1 =49B8
LI2 =49BB             LINKS =49B6
LOAD =FFD5            MODUS =0334
N0 =4889              N1 =48A1
N2 =489E              N3 =48B0
NETZ =4887            OB1 =49D3
OB2 =49DC             OBEN =49CF
P0 =48ED              P1 =48F9
P2 =48FD              P3 =48DD
PKTTAB =490C          PUNKT =48CE
PUNKT2 =48C8          RE1 =499E
RE2 =49A1             RECHTS =499C
SAVE =FFD8            SPEICH =4863
TASTE =0335           UN1 =49FB
UN2 =4A04             UNTEN =49F7
V =D000               XKOORD =0337
YKOORD =0336
//0000,4A1A,0A1A
```

Dieses Programm finden Sie natürlich auf der beigelieferten Diskette.

Sofern Sie beim Eingeben bzw. Laden keinen Fehler gemacht haben, meldet sich das vorliegende Programm wie folgt:

```
                 TITEL
==================================
                          Farbzutei-
                          lungsfeld/
                          Sekundär-
                          Operationen
   24x21-Feld
                          Original
                          feld 1
                          Original-
                          feld 2
```

Dieser Aufbau gibt Ihnen ständig die aktuellen Informationen, die Sie über den Zustand Ihres Arbeitssprites benötigen.

24x21-Feld:

In diesem übersichtlichen Arbeitsfeld können Sie mit Hilfe der verschiedenen Funktionen (s.u.) und dem Feld-Cursor Ihr Zeichen herstellen. Die Ziffern am linken Feldrand geben die y-Koordinate bzw. die Reihe, die Ziffern am oberen Rand die x-Koordinate oder das zuständige Bit im jeweiligen Byte des Spritespeichers an. Jedes gesetzte Bit (Punkt) wird als rotes Kästchen dargestellt.

Originalfelder:

Diese Felder stellen Ihr aktuelles Sprite in den jeweils gewählten Farben (s. "F") in der mit "G" gewählten Größe dar, um Ihnen einen anschaulichen Überblick über das spätere Aussehen des Sprites zu ermöglichen. Feld 1 zeigt dabei ständig den momentanen Zustand des Objektes in Normalgröße, Feld 2 dagegen in x- und/oder y-Richtung vergrößert an.

Farbzuteilungsfeld:

Hier wird Ihnen ein Überblick über die den Grundfarben zugeordneten Farben gegeben, die Sie beliebig ändern können (s.u.)

Sekundäroperationsfeld:

Hier werden weitere Informationen gegeben oder verlangt, sobald es dem Rechner notwendig erscheint.

Befehlssatz

Jeder Befehl besteht aus einem Tastendruck o h n e R e t u r n ! (Ausnahmen s.u.):

- B -

Einen kurzen Befehlsüberblick gibt Ihnen dieser Befehl. Nach Betätigung einer Taste kommen Sie zurück zur Spriteeingabe.

- <crsr> -

Mithilfe der Ihnen bekannten Cursorfunktionen können Sie den Feldcursor bewegen.

- 2/Q/W/A -

Zum leichteren Arbeiten besitzen diese Tasten ebenfalls Cursorfunktionen, deren Bewegungsrichtung aus der entsprechenden Tastaturposition resultiert.

- f1...f8 -

Mit diesen Tasten können Sie einen Punkt auf Ihr 24x21-Feld in der diesen Tasten (Grundfarben) zugeordneten Farbe zeichnen.

- F -

F versetzt Sie in die Lage, den Tasten f1-f8 (Grundfarben) entsprechende Farben zuzuordnen. Die mit diesen Tasten gezeichneten Punkte werden nun entsprechend in ihrem Originalfeld andersfarbig.
Achtung! Taste f1/2 (=Grundfarbe 0) steht für die Hintergrundfarbe!

- G -

Durch Drücken dieser Taste können Sie die Größe des Sprites, das sich in dem Originalfeld 2 befindet, wechseln.

- I -

Dieser Befehl ermöglicht Ihnen Ihr Sprite zu invertieren.

- V -

Sie können ihr Zeichen in Ihrem Feld beliebig verschieben. Dabei geht keine Information verloren, da die Ränder auf der anderen Seite wieder auftauchen.

Nach V drücken Sie R,L,O,U für Rechts, Links, Oben und Unten (Selbstverständlich läßt sich jede Operation direkt im Originalfeld beobachten).

- L -

Löscht Ihr gesamtes Sprite.

- C -

Bringt das Inhaltverzeichnis der Diskette (Direktory) auf den Bildschirm. Bei Drücken der ctrl.-Taste wird die Auflistung verlangsamt. Jede andere Taste bringt Sie wieder zurück.

- <ctrl>S / <ctrl>G -

Abspeichern (<ctrl>S) oder Laden (<ctrl>G) eines Sprites:
Geben Sie den Filenamen und - durch Semikolon(!) abgetrennt - die Gerätenummer an (bei Fehlen wird das zuletzt benutzte Gerät angesprochen). Nun drücken Sie <return>. Haben Sie diese Tasten versehentlich gedrückt, so genügt ein <return>, um wieder zurückzukehren (dies gilt für alle Sekundäroperationen, wie V,F,...). Beim Speichern wird ein Programmfile erzeugt, das entweder direkt in den Speicher eingeladen, oder auf die oben beschriebene Weise quasi wie ein Sequentielles File gehandhabt werden kann.

- <ctrl>X -

Sind Sie fertig und wollen beenden, so beantworten Sie auch diesem Befehl die Frage "Beenden??" mit J oder JA, löschen die restliche Zeile und drücken <return>. Jede andere Eingabe bringt Sie wieder zurück.

Ich hoffe, Sie werden viel Spaß haben mit Ihrer neuen Spritekreation, die Ihren Bekannten die Blässe ins Gesicht treiben wird.

5.3.2 Programmierung der Spriteeigenschaften

Das theoretische Wissen über die Art und Weise, wie ein Sprite definiert, eingeschaltet und seine Eigenschaften verändert werden, sollte Ihnen der Paragraph 4.5 vermittelt haben. Jetzt ist es an der Zeit, diese Dinge auch von BASIC aus anzuwenden. Dies soll anhand eines kleinen Beispielprogrammes geschehen, das sich fast aller Variationsmöglichkeiten bedient und schon einen kleinen Einblick in die typische Spriteanwendung zur Animation, also der bewegten Bilder, verschaffen soll. Dieses Thema soll zwar erst an späterer Stelle behandelt werden, doch bietet es sich hier geradezu an.

An diesem Beispielprogramm sollten Sie so viel ändern, wie sie wollen. Lediglich bei den Adressen der POKE-Befehle, also dem ersten Parameter, sollten Sie etwas vorsichtiger sein und zunächst einmal überlegen, welche Auswirkungen das haben könnte. Falls es Ihnen jedoch nichts ausmacht, den Rechner nach einem Absturz auszuschalten und das Programm neu zu laden, dann brauchen Sie sich aus dieser Warnung nichts zu machen. Doch hier zunächst einmal das Programm:

```
1000 REM ************************
1010 REM **                    **
1020 REM **  SPRITE-BEISPIEL   **
1030 REM **                    **
1040 REM ************************
1050 REM
1060 V = 53248 : REM BASISADRESSE VIDEOCONTROLLER
1070 POKE V+32,0 : POKE V+33,0 : REM RAHMEN UND HINTERGRUND = SCHWARZ
1080 REM
1090 REM SPRITEDEFINITION IN BLOCK 13:
1100 REM ******************************
1110 A1 = 13*64 : REM ADRESSE BLOCK 13
1120 FOR X=0 TO 62
1130 READ DT : POKE A1+X,DT : REM DATEN LESEN UND IN BLOCK 13 POKEN
1140 NEXT X
1150 REM DATAS ERSTES SPRITE:
1160 DATA 000,007,000
1170 DATA 056,013,128
1180 DATA 025,031,224
1190 DATA 126,031,204
1200 DATA 025,012,252
1210 DATA 056,007,000
1220 DATA 000,014,000
1230 DATA 000,014,000
1240 DATA 000,015,128
1250 DATA 000,014,192
1260 DATA 000,030,096
1270 DATA 000,062,048
1280 DATA 000,046,000
1290 DATA 000,079,000
1300 DATA 000,155,128
1310 DATA 001,025,192
1320 DATA 002,024,096
1330 DATA 003,024,096
1340 DATA 127,024,120
1350 DATA 188,024,000
1360 DATA 036,030,000
1370 REM
1380 REM SPRITEDEFINITION IN BLOCK 14:
1390 REM ******************************
1400 A2 = 14*64 : REM ADRESSE BLOCK 14
1410 FOR X=0 TO 62
1420 READ DT : POKE A2+X,DT : REM DATEN LESEN UND IN BLOCK 13 POKEN
1430 NEXT X
1440 REM DATAS ZWEITES SPRITE:
1450 DATA 000,003,010
1460 DATA 000,013,132
1470 DATA 058,031,224
1480 DATA 124,031,170
1490 DATA 058,015,250
1500 DATA 000,007,000
```

```
1510 DATA 000,014,000
1520 DATA 000,014,000
1530 DATA 000,014,000
1540 DATA 000,015,000
1550 DATA 000,015,128
1560 DATA 000,014,160
1570 DATA 000,030,000
1580 DATA 000,047,000
1590 DATA 000,077,128
1600 DATA 000,141,128
1610 DATA 001,057,128
1620 DATA 003,097,128
1630 DATA 127,113,128
1640 DATA 220,025,128
1650 DATA 036,001,224
1660 POKE 2046, A1/64 : REM ZUNAECHST BLOCK 13 FUER SPRITE 6
1670 POKE V+21, 2^6 : REM SPRITE 6 EINSCHALTEN
1680 POKE V+39+6, 1 : REM FARBE SPRITE 6: WEISS
1690 POKE V+27,   0 : REM SPRITE VOR HINTERGRUND
1700 POKE V+28,   0 : REM NORMALFARBEN-SPRITE
1710 POKE V+23, 2^6 : REM SPRITE 6 IN Y-RICHTUNG VERGROESSERT
1720 POKE V+29, 2^6 : REM SPRITE 6 IN X-RICHTUNG VERGROESSERT
1730 POKE V+2*6+1, 101: REM SPRITE 6 Y-KOORDINATE FESTLEGEN
1735 POKE V+2*6 , 100: REM SPRITE 6 X-KOORDINATE ZUR DEMONSTRATION
1740 POKE V+16,   0 : REM X-KOORDINATE HIGH-BIT LOESCHEN
1745 REM
1750 REM
1760 PRINT CHR$ (30) CHR$(147) : REM GRUEN + BILDSCHIRM LOESCHEN
1770 FOR X=1 TO 10
1780 PRINT : REM 10 LEERZEILEN
1790 NEXT X
1800 PRINT CHR$(18); : REM RVS ON
1810 FOR X=1 TO 40
1820 PRINT " "; : REM 40 INVERSE, GRUENE LEERZEICHEN
1830 NEXT X
1840 REM
1850 REM
1860 REM ***************
1870 REM **           **
1880 REM **  LAUFEN   **
1890 REM **           **
1900 REM ***************
1910 REM
1920 G = 10 : REM GESCHWINDIGKEIT
1930 F = 1 : REM FARBSTART
1940 FOR X=1 TO 400 STEP G
1950 F = F+.3: REM FARBE WECHSELN
1960 IF F=16 THEN F=1
1970 REM ---------------
1980 POKE V+39+6, F : REM FARBE WECHSELN
1990 POKE 2046, A1/64 : REM SPRITE 6 AUF BLOCK 13
```

```
2000 KO=X:IF X>255 THEN KO=X-256:POKE V+16,2^6:GOTO2020:REM HIGH B IT X-
KOORD. SETZEN
2010 POKE V+16, 0 : REM X-KOORDINATE - HIGH BIT LOESCHEN
2020 POKE V+2*6, KO : REM SPRITE BEWEGEN
2030 FOR Y=1 TO 40 : NEXT Y : REM WARTESCHLEIFE
2040 REM --------------
2050 POKE 2046, A2/64 : REM SPRITE 6 AUF BLOCK 14
2060 KO=X+G/2:IF KO>255 THEN KO=KO-256:POKE V+16,2^6:GOTO 2080:REM HIGH
BIT X-KOORD.
2070 POKE V+16, 0 : REM X-KOORDINATE - HIGH BIT LOESCHEN
2080 POKE V+2*6, KO : REM SPRITE BEWEGEN
2090 FOR Y=1 TO 40 : NEXT Y : REM WARTESCHLEIFE
2100 REM --------------
2110 NEXT X
```

In der ersten Zeile (Z. 1060) wird zunächst wieder die Variable V definiert, eine Prozedur, die Sie wohl bereits kennen und die vor jedem Ihrer Sprite- oder Graphikprogramme durchgeführt werden sollte. Alsdann ändern wir die Rahmen- und die Hintergrundfarbe in schwarz.

Jetzt beginnt die eigentliche Arbeit:

a) Spritedefinition:

Aus dem vorherigen Abschnitt kennen Sie bereits die ab Zeile 1100 folgende Routine: Die 63 Datenbytes, die ein Sprite definieren, werden aus nachstehenden DATA-Zeilen eingelesen und in den Speicher gePOKEt. Im ersten Fall wird die Definition in Block 13, im zweiten in Block 14 (ab Zeile 1400) eingeschrieben. Die Startadressen dieser beiden Blöcke wurden zunächst durch Multiplikation mit 64 errechnet und in die Speicher A1 und A2 abgelegt.

Wir haben nun also zwei Sprites definiert, die, wie Sie sehen werden, recht ähnlich aussehen. Es handelt sich dabei in beiden Fällen um einen älteren Herrn mit Pfeife, der gerade seinen Hund ausführt und von einem Vogel begleitet wird. Die beiden Sprites zeigen lediglich zwei Phasen der Bewegung unseres laufenden Männchens, der rauchenden Pfeife und des flatternden Vogels (Sie können sich die beiden Sprite getrennt einmal ansehen, wenn Sie in die "Laufroutine" nacheinander in die Zeilen 2030 und 2090 ein STOP einfügen). Unsere Absicht ist es nicht etwa, diese zwei Sprites gleichzeitig auf den Bildschirm zu bringen, wir wollen vielmehr versuchen, die sich nur durch jene paar Veränderungen unterscheidenden Objekte stets abwechselnd

so auf den Bildschirm zu bringen, daß sich aus den Unterschieden eine Bewegung entwickelt.

b) Blockvektor:

Wir verwenden für dieses Vorhaben Sprite 6 und legen als erstes den Block fest, aus dem die Spritedefinition für Sprite 6 stammen soll. Zu Beginn soll Block 13 die Definition liefern. Aus diesem Grunde müssen wir in das sechste der letzten 8 Bytes des Video-RAMs, also als Spritedefinitionsvektor, eine 13 POKEn (s. Kap. 4.5.3), was in Zeile 1660 geschieht.

c) Einschalten:

Doch damit haben wir längst noch nicht alles getan, um ein Sprite sichtbar zu machen. Wir müssen es zumindest noch einschalten. Dies wird in unserem Programm in Zeile 1670 unternommen. Hier wird in das Register 21 des VIC, das hierfür bekanntlich zuständig ist, der Wert 2^6 = 64 gePOKEt. Damit wird hier das 6. Bit gesetzt als Zeichen, daß nun Sprite 6 eingeschaltet ist.

d) Farbe:

Trotzdem werden wir höchstwahrscheinlich noch nicht viel zu sehen bekommen. Dies liegt meist daran, daß es noch außerhalb des Bildschirmfensters liegt. Darum werden wir uns später kümmern. Erst wollen wir die Farbe des Sprites festlegen. Hierfür sind bekanntlich die Register 39-46 des VIC zuständig. In Zeile 1680 wird in das sechste dieser 8 Register (also in Register 39+6 = 45) der Wert 1 für weiß (s. Anhang) gebracht, um diese Festlegung für das Sprite 6 vorzunehmen.

e) Priorität:

Wir entscheiden uns nun in Zeile 1690, ob wir unser Sprite vor oder hinter den Hintergrundzeichen sehen wollen. Mit anderen Worten: wird unser Sprite von dem später gezeichneten grünen Strich verdeckt oder verdeckt es diesen? Wir haben uns für den letzteren Fall entschieden und das entsprechende Bit im VIC-Register 27 gelöscht. Mit POKE V+27, 2^6 jedoch können Sie diesen Sachverhalt ändern. Probieren Sie einmal! Lesen Sie hierzu auch Kap. 4.5.4.3

f) Multicolor:

Wir haben unsere zwei Spritedefinitionen als Normalfarbensprites geplant und entwickelt. Daher müssen wir ebenfalls diesen Modus einschalten. Dieses Kriterium entscheidet sich, wie Sie sehen, bereits sehr früh, schon bei der eigentlichen Konstruktion.

g) Vergrößerung:

Und noch eine Entscheidung müssen wir treffen, bevor wir zur Tat schreiten können: Wollen wir das besagte Objekt in Originalgröße oder in irgendeine Richtung gedehnt anzeigen? Um die einzelnen Details der zwei Sprites besser erkennen zu können, wählten wir eine Vergrößerung des Objektes in x- und in y-Richtung und setzten in den Zeilen 1710/1720 die für Sprite 6 zuständigen Bits der Register 23 und 29 des VIC (s. Kap. 4.5.4.2). Doch auch in Originalgröße sieht das Ganze recht lustig aus (erreichbar durch Ersetzen des Wertes 2 hoch 6 durch 0). Versuchen Sie doch einmal eine Vergrößerung nur in x- oder y-Richtung!

h) Positionierung:

Endlich ist es soweit! Jetzt schreiten wir zur Tat und bringen unser Sprite sichtbar auf den Bildschirm. Nun nämlich legen wir die Koordinaten fest, bei denen sich das Objekt befinden soll. Später werden diese Angaben noch verändert, doch seien Sie hier schon der Vollständigkeit halber beigefügt. Wie Sie in Kap. 4.5.4.1 erfahren haben, sind hierfür die Register 0-16 zuständig, speziell für Sprite 6 die Nummern 2*6 = 12 (Low-Byte x-Koordinate), 2*6+1 = 13 (y-Koordinate) und 16 Bit 6 (High-Bit x-Koordinate). Diese Register bzw. das 6. Bit von Register 16 werden bei uns in den Zeilen 1730-1740 belegt.

In diesem Stadium schon ist unser Sprite sichtbar. Wir haben also alles Notwendige getan, um ein Sprite zu definieren. All diese Dinge sollten daher in jedem Spriteprogramm auftauchen. Sie sehen, daß es trotz des Komforts gar nicht so leicht ist, Sprites zu bedienen. Doch mit der Zeit werden Sie Routine bekommen und auch Wege finden, wie man die verschiedenen Dinge abkürzen oder gar weglassen kann.

Doch weiter bei unserer Programmbesprechung. Es folgt nun der Teil des Programms (ab Zeile 1750), der die vorherigen Dinge anwendet und für den eigentlich sichbaren Verlauf zuständig ist:

Nach dem Bildschirmlöschen und dem Zeichnen einer grünen Linie startet eine Schleife, die insgesamt 400 mal durchläuft und den Wert X von 1 bis 400 ansteigen läßt (Zeile 1940). Dabei entscheidet G über die Schrittgröße pro Schleifendurchlauf.

In dieser Schleife passiert eine ganze Menge: Zunächst wird der Speicher F um 0,3 erhöht (Zeile 1950). F wird verwendet, um die Farbe des Sprites in der Schleife zu verändern (Z. 1980). Da dabei nur ganze (integer-) Werte verwendet werden, ändert sich die Farbe nur dann, wenn sich der ganzzahlige Teil von F verändert. Dies geschieht somit etwa jede 3. Schleife.

Als Nächstes wird die Hauptaufgabe dieses Programmteils wahrgenommen: die Bewegung und der Definitionswechsel. Zunächst zum Definitionswechsel: Wie Sie wissen, ist für unser Sprite die Speicherstelle 2046 der Zeiger auf den jeweiligen Block, aus dem die Definition entnommen werden soll. Weiter oben hatten wir die beiden Objekte in den Blöcken 13 und 14 niedergelegt. Um nun eine Bewegung des Sprites an sich zu erhalten, müssen wir stets zwischen den beiden Definitionen hin und her schalten. An dieser Stelle (Z. 1990) wird deswegen der Wert 13 in 2046 gePOKEt, um den Vektor dorthin zu legen, es wird also das erste Sprite angezeigt. Weiter unten schließlich (Z. 2050) wird nach Definition 14 gewechselt, und das zweite Sprite ist sichtbar.

Um eine kontinuierliche Bewegung unseres Objektes erscheinen zu lassen (in Wahrheit wird es ja stets ruckweise verschoben, unser Auge jedoch nimmt dies als gleichmäßige Bewegung wahr), versetzen wir es jeden Schleifendurchlauf um einen oder mehrere Punkte in x-Richtung (die y-Koordinate bleibt konstant). Hierfür wird der ansteigende Speicher X verwendet. Er wird zunächst in KO (für KOordinate) zwischengespeichert (Z. 2000). Jetzt wird geprüft, ob dieser Wert größer ist als 255. Ist das der Fall, so reicht ein Byte alleine nicht mehr aus, und es muß gleichfalls noch das High-Bit der x-Koordinate in Register 16 gesetzt werden. Natürlich wird dabei der Speicher KO um 256 erniedrigt, um ein ILLEGAL QUANTITY ERROR zu verhindern. Nun wird auch das Low-Byte bestimmt (Z. 2020), und das Sprite ist versetzt. Im dritten Teil der Schleife wird diese Prozedur (nach einer Warteschleife) ein weiteres Mal ausgeführt (Z. 2060 ff.).

Im obigen Beispiel wurden Ihnen schon einige Techniken vermittelt, die Ihnen bei Ihrer Sprite-Programmierung behilflich sein werden. Selbstverständlich können Sie alle obigen Programme und Techniken gleichfalls auf die Supergraphik übertragen, wodurch sich Programmvereinfachungen und Geschwindigkeitsvorteile ergeben. Doch einige Themen wurden noch gar nicht angeschnitten: Multicolor, Kollisionsbehandlung und das Arbeiten mit mehreren Sprites gleichzeitig.

Dies soll im nächsten Schritt vorgenommen werden. Sie sollten inzwischen wissen, daß in Multicolor insgesamt 4 Farben pro Sprite verwendet werden können, unter der Einschränkung allerdings, daß hierbei die x-Auflösung um die Hälfte abnimmt, die Gesamtgröße jedoch konstant bleibt und damit die Punktbreite mit dem Faktor 2 zunimmt. Im Speicher wird jeder Punkt durch 2 Bits bestimmt, die das Register angeben, aus dem die jeweilige Farbe entnommen wird, angeben. Diese Register sind die Multicolor-Register 0 und 1 (VIC-Register 37/38) und das für jedes Sprite eigene Farb-Register (VIC-Register 39-46). Sind die beiden Bits gelöscht, so ist diese Stelle des Sprites durchsichtig (s. Kap. 4.5.3 f.).

Es können zwei verschiedene Kollisionen festgestellt werden: Sprite-Sprite (VIC-Register 30) und Sprite-Hintergrundzeichen (VIC-Register 31). In den jeweiligen Registern werden bei einer Berührung besagter Objekte die mit den Spritenummern korrespondierenden Bits gesetzt. Sollten Sie diese Dinge noch nicht kennen, so lesen Sie bitte unter Kap. 4.5.4.4 nach. In unserem Fall wird getestet, ob sich zwei Sprite berühren.

Auch hier soll wieder ein Beispielprogramm zur Veranschaulichung des weiter unten gesagten dienen, dessen Übertragung sich allein schon aufgrund der hübschen Spritedefinition lohnt:

```
100 REM *************************
110 REM **                     **
120 REM **  SPRITE-KOLLISION   **
130 REM **    (MULTICOLOR)     **
140 REM *************************
150 REM
160 V=53248 : REM VIC-BASISADRESSE
170 A1 = 11*64 : REM BLOCK 11
180 FOR X=0 TO 62
190 READ DT : POKE A1+X, DT : REM DEFINITION LESEN UND POKEN
200 NEXT X
210 POKE 2040, 11 : POKE 2042, 11 : REM BLOCKZEIGER SPRITE 0 UND 2 AUF 1
1
```

```
220 POKE V+28, 2^0 OR 2^2 : REM SPRITE 0 UND 2 AUF MULTICOLOR
230 POKE V+27, 2^0 : REM NUR SPRITE 2 HAT PRIORITAET VOR HINTERGRUND
240 POKE V+29, 2^0 OR 2^2 : POKE V+23, 2^0 OR 2^2 : REM BEIDE GROESSER
250 POKE V+21, 2^0 OR 2^2 : REM BEIDE SPRITES AN
260 POKE V+37, 7 : REM SPRITE-MULTICOLOR 0 = GELB
270 POKE V+38, 6 : REM SPRITE-MULTICOLOR 1 = BLAU
280 POKE V+39, 5 : POKE V+41, 8 : REM INDIVIDUALFARBEN SPRITE 0 UND 2
290 POKE V+16,0 : REM X-KOORD. HIGH-BITS LOESCHEN
300 POKE V+1,100 : POKE V+5,100 : REM Y-KOORDINATEN VORSETZEN
310 X2 = 255 : REM START-X-KOORDIANTE SPRITE 2
320 POKE V+0,0 : POKE V+4,X2 : REM X-KOORDIANTEN VORSETZEN
330 POKE V+30,0 : REM KOLLISION RUECKSETZEN
340 FOR X0=1 TO 255 : REM X-KOORD. SPRITE0
350 X2 = X2-1 : REM X-KOORD. SPRITE 2 ERNIEDRIGEN
360 POKE V+0, X0 : POKE V+4, X2 : REM X-KOORDIANTEN SPRITE 0/2 (LO W-
BYTES)
370 POKE V+1, 40*SIN(X0/30)+100 : REM BEWEGEN AUF SINUSKURVE
380 POKE V+5, 40*COS(X0/30)+100 : REM BEWEGEN AUF COSINUSKURVE
390 IF PEEK(V+30)=(2^0 OR 2^2) THEN GOTO 420
400 NEXT X0
410 REM
420 REM KOLLISIONSROUTINE:
430 REM
440 FOR X=1 TO 255
450 POKE V+39,X : POKE V+41,X : REM SPRITEFARBE WECHSELN
460 NEXT X
470 POKE V+0, 0 : REM SPRITES AUSEINANDERBRINGEN
480 POKE V+30,0 : REM KOLLISION RUECKSETZEN
490 REM
500 REM SPRITE-DATEN
510 REM
520 DATA 008,000,128
530 DATA 010,002,128
540 DATA 002,138,000
550 DATA 064,136,005
560 DATA 080,168,021
570 DATA 084,032,093
580 DATA 117,033,125
590 DATA 125,101,253
600 DATA 127,103,253
610 DATA 123,103,173
620 DATA 123,103,173
630 DATA 123,103,173
640 DATA 123,103,173
650 DATA 123,103,173
660 DATA 123,103,173
670 DATA 123,103,173
680 DATA 123,103,173
690 DATA 127,103,253
```

```
700 DATA 095,101,253
710 DATA 021,097,117
720 DATA 005,032,084
```

Zunächst werden hier wieder die üblichen Formalitäten zum Initialisieren der Sprites unternommen. In diesem Programm wird mit nur einer Spritedefinition gearbeitet, die in Block 11 untergebracht wird (Z. 170-200). Dafür aber sollen zwei Sprites gleichzeitig auf dem Bildschirm erscheinen. Wir wählen dafür Sprite 0 und 2. Beide Sprites besitzen jedoch dieselbe Form. Dies wird in Zeile 210 realisiert, in der beide Vektoren für Sprite 0 und 2 auf den Block 11 gerichtet werden. Beide Sprites also beziehen ihre Definition aus diesem Block. Trotzdem aber können Sie in den anderen Parametern weiterhin völlig unabhängig betrieben werden. Dies zeigen z.B. die Zeilen 230, in der beiden unterschiedliche Priorität vor den Hintergrundzeichen zugeschrieben wird, und 280, in der beide Sprite unterschiedliche Teilfarben erhalten. Wie gesagt wollen wir nun mit Multicolorsprites arbeiten. Die in den DATA-Zeilen angegebene Definition wurde deshalb als ein solches entworfen. Nun müssen wir dies gleichfalls dem VIC mitteilen, was in Zeile 220 geschieht. Hier haben wir auch gleich eine bisher noch nicht aufgetretene Schwierigkeit bewältigt. Bisher brauchten wir in allen Registern, die bitweise arbeiten lediglich ein Bit zu setzen. Hier aber wollen wir sowohl Sprite 0 als auch Sprite 2 zu Multicolorsprites erklären. Dies geschieht hier durch ODER- (OR-) Verknüpfung der beiden Einzelwerte 2hoch0 (Bit 0 gesetzt) und 2hoch2 (Bit 2 gesetzt). Sollten Sie damit noch Schwierigkeiten haben, so ziehen Sie sich doch einmal Kapitel 7.1 zu Gemüte.

Neues kommt auch in den Zeilen 260-280 auf uns zu. Hier werden die verschieden Farben der Multicolorsprites in die jeweiligen Register gelegt. Beachten Sie, daß nur Farbe 3 (Farbkanal 3) für alle Sprites unterschiedlich sein kann!

Unsere Sprites sind bereits eingeschaltet, aber wahrscheinlich noch nicht auf dem Bildschirm zu sehen. Wir werden sie also in das Bildschirmfenster hineinversetzen. Dies geschieht in den Zeilen 290-320. Eigentlich wäre diese Prozedur nicht notwendig, da alle Parameter (außer die High-Bits in Register 16) auch in der folgenden Bewegungsschleife gesetzt werden. Doch hier tritt noch eine weitere Schwierigkeit auf uns zu. Wir wollten die Sprites bei Ihrer Bewegung auf eventuelle Kollisionen überprüfen. Oft ist es aber so, daß sie sich bereits am Anfang berühren, was stets nach dem Einschalten des Rechners der Fall ist, da alle Koordinaten auf 0 stehen und auch Kollisionen vermerkt werden, wenn die Sprites außerhalb des Sicht-

bereiches sind. Wenn dies aber der Fall ist, dann würde unser Programm ja bereits ganz am Anfang eine Kollision feststellen und entsprechend verzweigen (s.u.). Wir bringen also die Objekte zunächst einmal auseinander.

Danach - und das ist ebenfalls sehr wichtig - setzen wir das Sprite-Sprite-Kollisionsregister zurück, indem wir den Wert 0 hineinschreiben, da dessen Inhalt bekanntlich so lange erhalten bleibt (genau genommen nur die gesetzten Bits), bis dieses Löschen vorgenommen wurde, auch wenn die Berührung längst nicht mehr besteht.

Danach können wir loslegen. Auch hier (wie in dem vorherigen Beispiel) werden die Sprites wieder in einer Schleife bewegt. Dabei wird der Schleifenzähler (X0) als laufend ansteigender Wert zur Bestimmung der x-Koordinate von Sprite 0 verwendet. Ein von 255 bis 1 absteigender Wert (X2), der stets einmal pro Schleife in Zeile 350 erniedrigt wird, legt die x-Koordinate des Sprite 2 fest (Zeile 360). Um nun Sprite 0 entlang einer Sinus- und Sprite 2 gemäß einer Cosinuskurve "fliegen" zu lassen, werden die y-Koordianten in den Zeilen 360 und 370 durch eine entsprechende Formel errechnet. Verändern Sie hierbei ruhig einmal die verschiedenen Parameter! (Die 40 sorgt für die Größe des Ausschlages (Amplitude), die 30 für die Länge der Kurven (Wellenlänge) und die 100 für die Verschiebung der gesamten Kurve in y-Richtung).

Nun kommen wir wieder zu einem sehr interessanten Teil: Die Kollisionsabfrage. In Zeile 390 wird gefragt, ob die Bits 0 und 2 des Registers 30, also des Sprite-Sprite-Kollisionsregisters, gesetzt sind. Ist das der Fall, so verzweigt das Programm nach Zeile 440 in die Kollisionsroutine. Hier wird ein kleiner Farb-Effekt erzeugt und die beiden sich berührenden Sprites auseinander gebracht, um das Kollisionsregister zu löschen. Weshalb wir die Sprite erst auseinander bringen müssen (wir hätten auch eines ausschalten können), ist weiter oben dargelegt. Damit hätten wir alles Notwendige, um Kollisionen zu bedienen.

Hier wurden Ihnen - ich hoffe recht anschaulich - die Grundlagen der Spriteprogrammierung vermittelt. Jetzt ist es an Ihnen, zu probieren und zu programmieren. Es gibt tausende von Anwendungen, denen lediglich Ihre Phantasie Grenzen setzt. Selbstverständlich kann in diesem Buch nicht auch nur ein Bruchteil dieser Dinge vermittelt werden. Es bedarf schon einiger Eigeninitiative, um hier Fuß zu fassen. Das aber ist ja gerade das Interessante am Programmieren. Der ganze Reiz wäre fort, wenn alle Möglichkeiten bereits vorgekaut vor

Ihnen lägen und Sie sie lediglich zusammenfügen müßten. So bleibt Ihnen genügend Freiraum, in dem Sie Ihrem Forschungsdrang freien Lauf lassen können.

5.4 Zeichensatzprogrammierung

Ein bisher recht stiefmütterlich behandeltes Thema ist das der Zeichensatzveränderung. Enorme Möglichkeiten tun sich hier auf. Ich kenne nur sehr wenige mehr oder minder anspruchsvolle Spiele, die nicht auf diese Eigenschaft Ihres Rechners zurückgreifen oder gar den Kern Ihres Inhalts nicht hierhin verlagern. Hier entscheiden die großen graphischen Möglichkeiten, die fast an die hochauflösende Graphik heranreichen, bei einer 8-9 mal schnelleren Verarbeitungsgeschwindigkeit. Mit der Fähigkeit, einen eigenen Zeichensatz zu kreieren wird der Commodore 64 ungeheuer anpassungsfähig. Der große Mangel vieler Computer, die sich mit den vielen verschiedenen Zeichensätzen oder Sonderzeichen in unterschiedlichen Ländern herumschlagen ist hier behoben. Nicht nur den amerikanischen, sondern auch den deutschen, schwedischen, ja sogar russischen, griechischen oder japanischen Zeichensatz kann Ihr Computer auf den Bildschirm bringen. Schriftarten können gewechselt werden und vieles mehr. Das bekannte Textomat beispielsweise, ein erfolgreiches Textverarbeitungsprogramm, bedient sich dieses inneren Schatzes. Sie sehen, diese Rechnereigenart ist nicht zu unterschätzen. Aus diesem Grunde widmen wir uns in diesem Kapitel der programmtechnischen Realisierung der Zeichensatzänderung. Diese ist nicht so einfach wie beispielsweise die Spriteprogrammierung und kommt zumindest an einer Stelle nicht ohne Maschinensprache aus, außer, Sie besitzen eine entsprechende BASIC-Erweiterung. Deshalb sollten Computerneulinge zunächst einmal diesen Abschnitt überspringen. Gleichfalls ist es empfehlenswert vor der Lektüre der folgenden Ausführungen den Paragraphen 4.6 gelesen und verstanden zu haben.

Wie Sie in dem Abschnitt 4.6 erfahren haben, besteht eine Zeichendefinition aus 8 Bytes zu je 8 Bits, was eine 8x8-Punkte-Matrix pro Zeichen ergibt. Gleichzeitig sind 512 verschiedene Zeichen speicherbar, maximal jedoch nur 256 zur selben Zeit auf dem Bildschirm darzustellen. Der gesamte Satz von 512 Zeichen besitzt eine Länge von 4 K und liegt normalerweise bei $D000-$DFFF (53248-57343) im sogenannten Zeichensatz-ROM. Er ist verschiebbar durch Änderung der Register 24 des VIC und Register 0, Bits 0/1 der CIA 2 (Adressen: $D018 = 53272 und $DD00 = 56576). Soweit das Wichtigste noch einmal in Kürze.

Bei der Zeichensatzerstellung muß man zwei Fälle unterscheiden. Im ersten Fall sollen nur einige wenige Teile des Zeichensatzes (Sonderzeichen) verändert werden. Bei der anderen Möglichkeit wird der gesamte Satz ausgewechselt. Zunächst zur ersten:

5.4.1 Änderung einiger Zeichen

Wollen wir ein paar Zeichen des Zeichengenerators durch eigene Zeichenformen ersetzen, so müssen wir den originalen Satz zunächst einmal aus dem ROM-Bereich in einen uns angenehmen RAM-Bereich kopieren. Anschließend ändern wir die Start-Adresse des Generators auf die in Abschnitt 4.3.2 (das Kapitel 4.3 sollten Sie, um alles richtig zu verstehen, bereits gelesen haben) erläuterte Art und Weise. Dann gehen wir hin und ändern die Zeichen, die wir ersetzen wollten.

Was hier so kurz und einfach erscheint, bedarf doch einigem an Hintergrundwissen und Programmiertechnik. Grundsätzlich kann man den Zeichengenerator nicht von BASIC aus auslesen bzw. kopieren. Dies ist dadurch verursacht, daß der Ort von $D000-$DFFF (53248-57343), in dem er sich befindet, normalerweise alle Eingabe/Ausgabe-Bereiche (E/A-Bereich) umfaßt und somit erst durch Ändern des Registers 1 der CPU (s. Kap. 4.5), also der Speicherstelle 1 eingeschaltet werden muß. Damit ist aber der E/A-Bereich, der von der Interruptroutine des Betriebssystems verwendet wird, lahmgelegt und es käme in BASIC zum Absturz des Computers. In Maschinensprache jedoch kann man den Interrupt durch Setzen des Interruptflags der CPU verhindern. Aus diesem Grunde geben wir Ihnen an dieser Stelle ein Maschinenprogramm an, das die Aufgabe des Kopierens und des Verschiebens des Zeichengenerators für Sie übernimmt. Letzteres könnte zwar ebensogut von BASIC aus geschehen, ist aber so einfacher.

```
 20:                     ;
 30:                     ;ZEICHENSATZVERSCHIEBUNG:
 40:                     ;*************************
 50:                     ;
 60: C800                      *=   $C800    ;STARTADRESSE (51200)
 70: C800           V          =    53248    ;BASISADRESSE VIDEOCONTROLLER
 80: C800           SATZ       =    53248    ;BASISADRESSE ZEICHENSATZ
 90: C800           ZIEL       =    $3000    ;BASISADRESSE KOPIERADRESSE
100: C800 78        START SEI              ;INTERRUPT VERHINDERN
110: C801 A5 01           LDA  $01        ;CPU-REG. 1
120: C803 48              PHA             ;RETTEN
130: C804 29 FB           AND  #%11111011  ;BIT 2 LOESCHEN
140: C806 85 01           STA  $01        ;(ZEICHENGENERATOR AUSLESBAR)
150: C808 A9 D0           LDA  #>SATZ
160: C80A 85 03           STA  $03        ;QUELLADRESSE HIGH-BYTE
170: C80C A9 30           LDA  #>ZIEL
180: C80E 85 05           STA  $05        ;ZIELADRESSE HIGH-BYTE
190: C810 A0 00           LDY  #$00
200: C812 84 02           STY  $02
210: C814 84 04           STY  $04        ;LOW-BYTES = 00
220: C816 A2 20           LDX  #$20       ;ZAEHLER FUER 4 K-BYTE
230: C818 B1 02     KOPIE LDA  ($02),Y ;BYTE LADEN
240: C81A 91 04           STA  ($04),Y ;UND KOPIEREN
250: C81C C8              INY
260: C81D D0 F9           BNE  KOPIE     ;256 MAL
270: C81F E6 03           INC  $03
280: C821 E6 05           INC  $05        ;HIGH-BYTES ERHOEHEN
290: C823 CA              DEX
300: C824 D0 F2           BNE  KOPIE     ;4 K KOPIEREN
310: C826 68              PLA
320: C827 85 01           STA  $01        ;ZEICHENGENERATOR AUS-E/A EIN
330: C829 AD 18 D0        LDA  V+24       ;ZEICHENGENERATORADRESSE
340: C82C 29 F1           AND  #%11110001
350: C82E 09 0C           ORA  #%00001100
360: C830 8D 18 D0        STA  V+24       ;AUF $3000 SETZEN
370: C833 58              CLI             ;INTERRUPT WIEDER ERLAUBEN
380: C834 60              RTS             ;ZURUECK NACH BASIC
```

Und hier der dazugehörige BASIC-Lader:

```
100 FOR I = 51200 TO 51252
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,165,  1, 72, 41,251,133,  1,169,208,133,  3
130 DATA 169, 48,133,  5,160,  0,132,  2,132,  4,162, 32
140 DATA 177,  2,145,  4,200,208,249,230,  3,230,  5,202
150 DATA 208,242,104,133,  1,173, 24,208, 41,241,  9, 12
160 DATA 141, 24,208, 88, 96
170 IF S <> 5884 THEN PRINT "FEHLER IN DATAS !!" : END
180 PRINT "OK"
```

Wollen Sie also Teile Ihres Zeichensatzes ändern, so brauchen Sie lediglich diesen Lader einzuladen, RUN zu drücken und anschließend ein SYS 51200 einzutippen, und schon ist der originale Zeichensatz verschoben (Sie können, falls Sie einen Monitor besitzen, natürlich auch direkt das Maschinenprogramm eingeben, abspeichern und später mit LOAD "name",8,1 einladen).

Der ursprüngliche Grund für diese Exkursion war aber das Ziel, verschiedene Zeichen des Zeichengenerators zu verändern. Dies wollen wir jetzt unternehmen.

Aus dem Abschnitt 4.6 kennen wir bereits den 8x8-Aufbau (bzw. 4x8 bei Multicolor) eines Zeichens und seine Abspeicherung in 8 Bytes. Um nun eine Zeichendefiniton auszuwechseln, müssen wir uns zunächst eine eigene überlegen. Dies geschieht am besten auf die gleiche Weise, wie bei den Sprites mit Hilfe eines Zeichenentwurfsblattes, das Sie ebenfalls im Anhang finden. Die Anwendung kennen Sie bereits von den Sprites her. Jede Reihe dieses Zeichenentwurfes bildet dabei genau ein Byte, jeder gesetzte Punkt ein Bit der Definition. Nun haben wir beispielsweise folgendes Gebilde creiert:

```
   Bit:  7 6 5 4 3 2 1 0
Byte 0:  . . . * * * . .
Byte 1:  . * * * . . . .
Byte 2:  * * . . * * * *
Byte 3:  * * . . . . . .
Byte 4:  * * . . * * * *
Byte 5:  . * * * . . . .
Byte 6:  . . . * * * . .
Byte 7:  . . . . . . . .
```

Wir erhalten damit folgende 8 Bytes:

```
Byte 0: %0001 1100 = $1C = 028
Byte 1: %0111 0000 = $70 = 112
Byte 2: %1100 1111 = $CF = 207
Byte 3: %1100 0000 = $C0 = 192
Byte 4: %1100 1111 = $CF = 207
Byte 5: %0111 0000 = $70 = 112
Byte 6: %0001 1100 = $1C = 028
Byte 7: %0000 0000 = $00 = 000
```

Damit taucht auch gleich die nächste Frage auf: Welches Zeichen soll unsere Neuschöpfung ersetzen, und welchem der insgesamt 4 Zeichensätze (s. Kap. 4.6) soll es angehören? In unserem Fall wollen wir statt des normalen Pfundzeichens, das im Groß/Graphikmodus erreichbar ist, nun diese Definition setzen.

Jetzt müssen wir feststellen, wo die entsprechenden 8 Bytes innerhalb des gesamten Zeichengenerators liegen. Dazu verwenden wir die im 4. Kapitel angegebene Formel:

adresse = basisadresse + 8 * bildschirmcode

Die Basisadresse des Zeichengenerators hängt grundsätzlich von dem Bereich ab, in den wir diesen verschoben haben. Da wir uns in unserem Beispiel innerhalb der Adressen $3000-$3FFF (12288-16383) befinden und wir nur den Großschrift/Graphikzeichenmodus verändern wollen (die Definitionen für dessen Zeichen liegen von $3000-$37FF (12288-14335)), liegt dieser erste Wert hier schon einmal fest. Bleibt lediglich der Bildschirmcode: Diesen erfahren wir durch Nachschlagen in der Bildschirmcodetabelle im Anhang. Er ergibt sich für das normale Pfundzeichen zu 28 ($1C) (das inverse Pfundzeichen hätte den Code: 128+28=156). Damit können wir die verschiedenen Werte in unsere Formel einsetzen:

adresse = 12288 + 8*28 = 12512 = $30E0

Damit wissen wir, daß die 8 Bytes von $30E0 bis $30E7 (12512-12519) die Definition für besagtes Pfundzeichen enthalten. Mit einem Speichern der obigen 8 Bytes unseres Sonderzeichens in diese Positionen ändern wir sofort das Aussehen aller Pfundzeichen auf dem Bildschirm. Dies geschieht etwa durch folgendes BASIC-Programm (selbstverständlich, nachdem Sie zunächst mit obiger Routine den Zeichensatz kopiert und verschoben haben):

```
10 REM ZEICHENAENDERUNG
20 FOR X=0 TO 7
30  READ DT : REM 8 DATEN LESEN
40  POKE 12288 + 8*28 + X, DT : REM UND EINPOKEN
50 NEXT X
60 DATA 28,112,207,192,207,112,28,0
```

Bevor Sie dieses Programm ablaufen lassen, sollten Sie ein paar Pfundzeichen auf den Bildschirm bringen, um die Wirkung direkt zu sehen. Wenn Sie nach dem Starten dann auf den Klein-/Großschrift-Modus umschalten (mit <shift><C=>), dann sehen Sie, daß das Pfundzeichen dieses Satzes aus einem anderen Speicherbereich stammt. Diese Operation können Sie natürlich mit allen anderen Zeichen genauso durchführen und sich damit ein ganzes Reservoir an Sonderzeichen schaffen, die bei der Erstellung von besonders schönen und abwechslungsreichen Graphiken oder beim kommerziellen Gebrauch Verwendung finden.

5.4.2 Änderung eines Zeichensatzes

Wollen wir einen ganzen Zeichensatz auswechseln, so brauchen wir den originalen nicht extra zu kopieren. Damit kommt man völlig mit den Möglichkeiten, die in BASIC bestehen aus. Man läd einfach den neuen Zeichensatz in einen bestimmten Speicherbereich ein und teilt dem VIC mit, daß er die Definition der verschiedenen Buchstaben etc. von nun ab von dort holen soll.

Doch zunächst einmal muß dieser neue Zeichensatz entstehen, der jetzt von mir aus alle griechischen, russischen, kyrillischen Zeichen oder nur Blocksatz, Elite oder Schreibschrift enthält. Dies ist normalerweise aufgrund der Vielzahl der verschiedenen Zeichen recht mühselig und mancher Programmierer würde ob dieser schier unbewältigbar erscheinenden Arbeit verzweifeln. Jedes Zeichen müßte auf einem Blatt entworfen, in Bytes übersetzt und schließlich entweder per Monitor direkt oder - was noch mühseliger wäre - von BASIC aus durch POKEs in den Speicher eingegeben werden. Weil aus diesen Gründen ein vernünftiges Arbeiten mit ganzen Zeichensätzen kaum realisierbar erscheint, werden wir Ihnen an dieser Stelle ein Programm vorstellen - ähnlich dem Spriteeditor -, mit dem Sie auf einfachste Art und Weise solche Tätigkeiten vornehmen können. Auch hier erwartet Sie wieder einiges an Tipparbeit, die sich jedoch ernsthaft lohnt! Halten Sie dies

jedoch für verlorene Zeit, so weise ich Sie auch hier wieder darauf hin, daß Sie exklusiv zu diesem Buch eine Diskette mit allen hier aufgeführten Programmen erhalten können. Speziell zum Zeichenformer ist dort auch eine Version für Multicolorzeichen und ein Programmteil "Erläuterungen" vorhanden.

In dem vorliegenden Programm[1] wurde wieder viel Wert auf Dokumentation gelegt, damit Sie die Möglichkeit haben, dieses Programm auch zu verstehen.

```
10 REM **********************
20 REM **                  **
30 REM **  ZEICHENFORMER   **
40 REM **                  **
50 REM **********************
60 REM
70 REM INITIALISIERUNG:
80 REM *****************
90 GOSUB 10000 : REM MASCHINENROUTINEN EINLESEN
100 POKE 53280,0:POKE 53281,0:REM HINTERGRUND-/RAHMENFARBE
110 POKE 763,0:POKE 650,255:REM MODUS=0:ALLE ZEICHEN REPEAT
120 POKE 45,0:POKE 46,80:RUN 130:REM BASICENDE=$5000
130 REM
140 REM MASCHINENROUTINEN:
150 REM ******************
160 IN%=18432:REM INITROUTINE
170 PU%=18633:REM PUNKT EINZEICHNEN
180 NE%=18586:REM KOORDINATENSYSTEM
190 LA%=18487:REM ZEICHENSATZ LADEN
200 SP%=18515:REM ZEICHENSATZ SPEICHERN
210 CA%=18765:REM CATALOG
220 BE%=18689:REM BEFEHLSIDENTIFIZIERUNG
230 Q  =13048:REM TESTZEICHENADRESSE
240 REM
250 REM CONTROLZEICHEN:
260 REM ****************
270 C0$=CHR$(147):REM BILDSCHIRM LOESCHEN
280 C1$=CHR$( 19):REM HOME
290 C2$=CHR$(183):REM HOCHSTRICH
300 C3$=CHR$(117)+CHR$( 99)+CHR$(105):REM OBERER ZEICHENFENSTERRAND
310 C4$=CHR$(106)+CHR$( 99)+CHR$(107):REM UNTERER RAND
320 C5$=CHR$( 98):REM MITTELSTRICH (SENKR)
330 C6$=CHR$( 18):REM RVS ON
340 C7$=CHR$(146):REM RVS OFF
350 NA%=704:REM FILENAMENLAENGE($02C0)
360 GA%=186:REM GERAETEADRESSE($BA)
```

1 Dieses Programm funktioniert nur ohne die Supergraphik

```
370 MO%=763:REM MODUS
380 TA%=764:REM TASTE/BEFEHLSCODE
390 YK%=765:REM Y-KOORD
395 XK%=766:REM X-KOORD
400 REM
410 REM FARBEN DEFINIEREN:
420 REM ******************
430 DATA 144,  5, 28,159,156, 30, 31,158
440 DATA 129,149,150,151,152,153,154,155
450 DIM C$(16):FOR Y=0 TO 15:READ X:C$(Y)=CHR$(X):NEXT Y
460 N=1:F(0)=0:F(1)=1:V$=" ":SYS IN%:REM FARBEN/INIT
500 REM
510 REM LOESCHROUTINE (FELDAUFBAU):
520 REM ****************************
530 FOR Y=Q TO Q+7:POKE Y,0:NEXT Y:REM TESTZEICHEN LOESCHEN
540 PRINT C0$
550 PRINT C1$;SPC(10);C$(7);"ZEICHEN-NEU-CREATION"
560 PRINT SPC(11);C$(1);"(C) BY AXEL PLENGE"
570 PRINT C$(4);:FOR X=1 TO 40:PRINT C2$;:NEXT X:PRINT C$(6):PRINT
580 PRINT "   76543210":SYS NE%
590 PRINT "   ";:FOR X=0 TO 7:PRINTC2$;:NEXT X
600 A=15:B=5:GOSUB 9000:REM POSITIONIEREN
610 PRINT C3$:PRINT TAB(15);C5$;CHR$(127);C5$:POKE 55552,F(1):REM TESTZE
ICHEN MIT FARBE
620 PRINTTAB(15);C4$:GOSUB 3000:X=0:Y=0:REM STATUSFELD/X/Y-KOORD=0
700 REM
710 REM EINGABESCHLEIFE:
720 REM ****************
730 A=X+3:B=Y+6:GOSUB 9000:REM POSITIONIEREN
740 POKE XK%,X:POKE YK%,Y:F=0:REM KOORDINATEN UEBERGEBEN
750 PRINT C$(7);C6$;" ";CHR$(157);:REM BLINKPHASE AN
760 FOR S=1 TO 50:GETA$:IF A$<>"" THEN 800
770 NEXT S:SYS PU%:FOR S=1 TO 50:GET A$:IF A$="" THEN NEXT S:GOTO 750:RE
M AUSSCHALTEN
800 REM
810 REM BEFEHLSERKENNUNG:
820 REM *****************
830  SYS  PU%:C=ASC(A$):POKE  TA%,C:SYS  BE%:S=PEEK(TA%):REM  BEF-UEBER
GABE/RUECKMELDUNG
840 REM VERTEILUNG:
850 ON S GOTO 1600,2800,2900,1060,1060,1080,1080,1100,1100
860 ON S-9 GOTO 1110,1110,3100,3100,3100,3100
870 ON S-15 GOTO 3100,3100,3100,3100,500,1700
880 ON S-21 GOTO 1800,1900,2000,2100,1200,3400
890 ON S-27 GOTO 1300,3200,1400,700
1000 REM
1010 REM BEFEHLSBEARBEITUNG:
1020 REM *******************
1030 REM
1040 REM CURSORBEWEGUNG:
1050 REM ***************
```

```
1060 X=X+1:IF X=8 THEN X=0:GOTO 1100
1070 GOTO 700:REM RECHTS
1080 X=X-1:IF X<0 THEN X=7:GOTO 1110
1090 GOTO 700:REM LINKS
1100 Y=(Y+1)AND7:GOTO 700:REM RUNTER
1110 Y=(Y-1)AND7:GOTO 700:REM HOCH
1200 REM
1210 REM BEENDEN:
1220 REM ********
1230 A=2:B=15:GOSUB 9000:REM POSITIONIEREN
1240 PRINT C6$;C$(7);"BEENDEN?";C7$;C$(6):INPUT T$
1250 IF T$="J" OR T$="JA" THEN SYS 64738:REM KALTSTART
1260 GOTO 540
1300 REM
1310 REM CATALOG:
1320 REM ********
1330 PRINT C0$:SYS CA%:GOSUB 9100:GOTO 540
1400 REM
1410 REM VERSCHIEBUNG:
1420 REM *************
1430 GOSUB 8999:PRINT C$(1);"VERSCHIEBUNG:":REM MELDEFELD
1440 PRINT "NACH: RECHTS(R), LINKS(L),":PRINT SPC(6) "OBEN  (O), UNTEN(U
):"
1450 GOSUB 9100:IF T$<>"R" THEN 1470
1460 FOR T=0 TO 7:R=PEEK(Q+T):POKE Q+T,(R/2) + (R AND 1)*128:NEXT T:REM
RECHTS
1470 IF T$<>"L" THEN 1490
1480 FOR T=0 TO 7:R=PEEK(Q+T)*2:POKE Q+T,(R AND 255) + R/256:NEXT T:REM
LINKS
1490 S=1:IF T$="O" THEN 1510
1500 S=-1:IF T$<>"U" THEN 1520
1510 FORT=0TO7:P(T)=PEEK((7ANDT+S)+Q):NEXTT:FORT=0TO7:POKEQ+T,P(T):NEXTT
:REM HOCH/RUNTER
1520 GOSUB 9200:GOTO 550
1600 REM
1620 REM MODUSWECHSEL:
1630 REM *************
1640 M=ABS(M-1):POKE MO%,M:GOSUB 3000:GOTO 700
1700 REM
1710 REM ZEICHEN DEFINIEREN:
1720 REM *******************
1730 GOSUB 8999:PRINT C6$;C$(5);"GEBEN SIE AN:";C7$
1740 PRINT C$(7);"ZUORDNUNG DES ZEICHENS: ";C$(6);CHR$(127);C$(7)
1750 GOSUB 2500:IF F=1 THEN F=0:GOTO 540:REM ADRESSENERR.+FEHLERABFR.
1760 FOR X=0 TO 7:POKE T+X,PEEK(Q+X):NEXT X:REM SPEICHERN
1770 FOR X=1 TO 1500:NEXT X:GOSUB 9200:GOTO550:REM WARTEN+LOESCHEN
1800 REM
1810 REM ZEICHEN HOLEN:
1820 REM **************
1830 GOSUB 8999:PRINT C$(1);"GEBEN SIE DAS ZU BEARBEITENDE":PRINT "ZEICH
EN EIN:"
```

```
1840 GOSUB 2500:IF F=1 THEN F=0:GOTO 540:REM ADRESSENERR.+FEHLERABFR.
1850 FOR Y=0 TO 7: POKE Q+Y,PEEK(T+Y):NEXT Y:GOSUB 9200:GOTO 550:REM LAD
EN
1900 REM
1910 REM ZEICHEN INVERTIEREN:
1920 REM ********************
1930 FOR B=0 TO 7:POKE Q+B,255-PEEK(Q+B):NEXT B:GOTO 550
2000 REM
2010 REM ZEICHENSATZ SPEICHERN:
2020 REM **********************
2030 GOSUB 8999:PRINT C6$;C$(1);"ZEICHENSATZABSPEICHERUNG";C7$
2040 GOSUB 2300:IF F=1 THEN F=0:GOTO 2000:REM EINGABE/FEHLERABFR.
2050 IF F=2 THEN F=0:GOTO 2150:REM FEHLER
2060 SYS SP%:GOTO 2200:REM SPEICHERN
2100 REM
2110 REM ZEICHENSATZ LADEN:
2120 REM ******************
2130 GOSUB 8999:PRINT C6$;C$(1);"ZEICHENSATZ LADEN:";C7$:
2140 GOSUB 2300:IF F=1 THEN F=0:GOTO 2100
2150 IF F=2 THEN F=0:GOTO 540
2160 SYS LA%
2200 REM FEHLERABFRAGE (NUR FUER DISK!):
2210 OPEN 1,8,15:INPUT#1,DS,DS$,DT,DB:CLOSE1
2220 IF DS<20 THEN 500:REM OK
2230 PRINT:T$=STR$(DS)+","+DS$+","+STR$(DT)+","+STR$(DB)
2240 GOSUB 9310:PRINT T$:FOR S=1 TO 2000:NEXT S:GOTO 500:REM BLINKEN
2300 REM
2310 REM NAMENEINGABE:
2320 REM *************
2330 A$="":PRINT:PRINT "FILENAME"+C$(6);:INPUT A$:T=LEN(A$)
2340 S=VAL(RIGHT$(A$,1))
2350 IF S<>0 AND LEFT$(RIGHT$(A$,2),1)=";" THEN T=T-2:POKE GA,S:RE M
GERAETEADR.
2360 IF T=0 THEN F=2:RETURN:REM KEIN NAME
2370 IF T>17 THEN 2390
2375 REM NAMEN AN MASCHINENROUTINEN (704=$02C0):
2380 POKE NA%,T:FOR S=1 TO T:POKE NA%+S,ASC(MID$(A$,S,1)):NEXT S:RETURN
2390 PRINT CHR$(145);:T$=C6$+"LAENGE!"+C7$:GOSUB 9310:REM FEHLERMELDUNG
2400 PRINT C$(6):F=1:RETURN
2500 REM
2510 REM ZEICHENADRESSE ERRECHNEN:
2520 REM *************************
2530 PRINT "ZEICHENSATZ(1-4): ";N:A$=""
2540 PRINT "TASTE(F3) ODER ASCII(F5)?":GOSUB 9100
2550 ON ABS(ASC(T$)-132) GOTO 2570,2570,2590,2590
2560 GOTO 9300
2570 INPUT "TASTE";A$:A$=LEFT$(A$,1):IF A$="" THEN 9300:REM TASTENEINGAB
E
2580 T=ASC(A$):GOTO 2620
2590 INPUT "ASCII";A$:IF A$="" THEN 9300:REM ASCII-EINGABE
2600 T=VAL(A$)/255:T=INT((T-INT(T))*255):A$=CHR$(T)
```

```
2610 IF T<32 OR (T<160 AND T>127) THEN 9300
2620 IF T>191 THEN T=T-96:REM ASCII-UMWANDLUNG
2630 PRINT CHR$(145);"TASTE:";A$;" ASCII:";T;:IF B>8 THEN V$=A$
2640 REM UMRECHNUNG (T IST DER NORMALE ASCII-WERT DES ZEICHENS):
2650 IF T<64  THEN S=256:GOTO 2680
2660 IF T<128 THEN S=256*((32 AND T)/16)
2670 IF T>159 THEN S=256*(INT(T/32)-2)
2680 T=(N+47)*1024+S+(31 AND T)*8:RETURN
2800 REM
2810 REM ZEICHENSATZWECHSEL:
2820 REM *******************
2830 A=36:B=5:GOSUB 9000:PRINT C$(2);C6$;N;C7$;C$(6):REM NUMMER INVERTIE
REN
2840 GOSUB 9100:S=VAL(T$):IF S=0 OR S>4 THEN S=N:REM S=NEUER/N=ALTER ZEI
CHENSATZ
2850 N=S:GOSUB 3000:GOTO 700
2900 REM
2910 REM ZEICHEN HOLEN (IN MODUS 1):
2920 REM ****************************
2930 IF C<32 OR (C>127 AND C<160) THEN 700
2940 T=C:R=N:V$=A$:GOSUB 3000:GOTO 1850
3000 REM
3010 REM STATUSFELD ERSTELLEN:
3020 REM *********************
3030 A=20:B=5:GOSUB 9000:PRINT C$(7);"MODUS:";M;"/ SATZ:";N:A$=V$:T=ASC(
A$)
3040 PRINT TAB(20);CHR$(17);CHR$(17);C$(6);
3050 GOSUB 2620:PRINT CHR$(157);" ":PRINT
3060 PRINT TAB(20);C$(7);"FARBZUTEILUNG:"
3070 PRINT TAB(20);C$(2);:FOR S=1 TO 14:PRINTCHR$(163);:NEXT S:PRINT C$(
6)
3080 FOR S=0 TO 1:PRINT TAB(20);"GRUNDFARBE";S;":";F(S):NEXT S:RETURN
3100 REM
3110 REM PLOT:
3120 REM *****
3130 S=((S-12) AND 2)/2:REM PLOTFARBE FESTSTELLEN
3140 T=2^(7-X):POKE Q+Y,PEEK(Q+Y) AND (255-T) OR S*T:GOTO 700
3200 REM
3210 REM FARBENWAHL:
3220 REM ***********
3230 GOSUB 8999:PRINT TAB(4);C$(1)"F"C$(2)"A"C$(3)"R"C$(4)"B"C$(5)"E"C$(
6)"N";
3240 PRINT C$(7)"W"C$(4)"A"C$(6)"H"C$(2)"L"C$(7)":"
3250 PRINT TAB(4);C$(1);CHR$(172);:FOR S=1 TO 32:PRINT CHR$(162);:NEXT S
:PRINT CHR$(187)
3260 FOR S=1 TO 2:PRINT TAB(4);C6$;CHR$(161);
3270 FOR T=0 TO 15:PRINT C$(T);"  ";:NEXT T:PRINT C$(1);C7$;CHR$(161):NE
XT S
3280 PRINT TAB(4);C6$;CHR$(161);" 0 1 2 3 4 5 6 7 8 9101112131415";C7$;C
HR$(161)
3290 PRINT:PRINT C$(6);"    FUER GRUNDFARBENNR.(F1/F3): ";
```

```
3300 GOSUB 9100:T=ASC(T$)-133:REM FUNKTIONSTASTE
3310 IF T<0 OR T>1 THEN GOSUB 9300:GOTO 540:REM FEHLER
3320 IF T>1 THEN T=T-4
3330 PRINT T:T$="":INPUT "    FARBE ";T$:S=ABS(INT(VAL(T$)))
3340 IF T$="" OR S>15 THEN GOSUB 9300:GOTO540:REM FEHLER
3350 F(T)=S:POKE 53281+T,S:REM FARBE SETZEN
3360 GOTO 540
3400 REM
3410 REM BEFEHLSSATZ:
3420 REM ************
3430 PRINT C0$;C6$;C$(2)"              ";C$(7);
3440 PRINT "BEFEHLSSATZ";C$(2);"              ";C7$;
3450 PRINT C$(4);:FOR S=1 TO 40:PRINT CHR$(184);:NEXT S:PRINT
3460 PRINT C$(1)" NR.  "C6$"BEFEHL "C7$"-"C$(5)" FUNKTION"C$(4)
3470 FOR S=1 TO 10:PRINT "----";:NEXT S
3480 PRINT C$(1)" (1)  "C6$"(<>^..)"C7$"-"C$(5)" CURSORBEWEGUNGEN"
3490 PRINT C$(1)" (2)  "C6$"(<DEL>)"C7$"-"C$(5)" MODUSWECHSEL"
3500 PRINT C$(1)" (3)  "C6$"(CTRL^)"C7$"-"C$(5)" ZEICHENSATZWECHSE L"
3510 PRINT C$(1)" (4)  "C6$"(F1-F8)"C7$"-"C$(5)" PLOT IN FARBEN 0-15"
3520 PRINT C$(1)" (5)  "C6$"(F)    "C7$"-"C$(5)" FARBEN 0-15 F. F1 /3
DEF."
3530 PRINT C$(1)" (6)  "C6$"(B)    "C7$"-"C$(5)" BEFEHLSSATZ"
3540 PRINT C$(1)" (7)  "C6$"(D)    "C7$"-"C$(5)" ZEICHEN DEFINIERE N"
3550 PRINT C$(1)" (8)  "C6$"(H)    "C7$"-"C$(5)" ZEICHEN HOLEN"
3560 PRINT C$(1)" (9)  "C6$"(I)    "C7$"-"C$(5)" ZEICHEN INVERTIER EN"
3570 PRINT C$(1)"(10)  "C6$"(V)    "C7$"-"C$(5)" ZEICHEN VERSCHIEB EN"
3580 PRINT C$(1)"(11)  "C6$"(L)    "C7$"-"C$(5)" ZEICHEN LOESCHEN"
3590 PRINT C$(1)"(12)  "C6$"(CTRLG)"C7$"-"C$(5)" GET-ZEICHENS. LAD EN"
3600 PRINT C$(1)"(13)  "C6$"(CTRLS)"C7$"-"C$(5)" SAVE-ZEICHENS. SP
EICHERN"
3610 PRINTC$(1)"(14)  "C6$"(C)    "C7$"-"C$(5)" DIREKTORY/CATALOG"
3620 PRINT C$(1)"(15)  "C6$"(CTRLX)"C7$"-"C$(5)" BEENDEN"
3630 GOSUB 9100:GOTO 540
8000 REM
8100 REM UNTERPROGRAMME:
8200 REM ***************
8300 REM
8400 REM POSITIONIERUNG:
8500 REM ***************
8999 A=0:B=16:REM MELDEFELD
9000 PRINT C1$;:FOR S=2 TO B:PRINT:NEXT S:PRINT TAB(A);:RETURN
9050 REM
9060 REM TASTENEINGABE:
9070 REM **************
9100 WAIT 198,255:GET T$:RETURN
9150 REM
9160 REM MELDEFELD LOESCHEN:
9170 REM ******************
9200 A=0:B=16:GOSUB 9000:FOR S=1 TO 5:GOSUB 9210:PRINT:NEXT S:RETURN
9210 FOR T=1 TO 9:PRINT "    ";:NEXT T:RETURN
9250 REM
```

```
9260 REM FEHLERBLINKEN:
9270 REM **************
9300 T$="UNZULAESSIG!"
9310 PRINT C$(1):FOR S=1 TO 9:PRINT T$:GOSUB 9330:PRINT CHR$(145);
9320 GOSUB 9210:PRINT CHR$(145):GOSUB 9330:NEXT S:GOSUB 9200:F=1:RETURN
9330 FOR T=1 TO 75:NEXT T:RETURN:REM WARTESCHLEIFE
9890 REM
9900 REM *************************
9910 REM **                     **
9920 REM **  MASCHINENROUTINEN  **
9930 REM **                     **
9940 REM *************************
9950 REM
9960 REM DATAS WERDEN NACH DEM STARTEN GELOESCHT !!!
9970 REM
10000 FOR I = 1 TO 16 : READ X : NEXT I : REM VORDERE DATAS UEBERSPRINGE
N (FARBEN)
10005 FOR I = 18432 TO 18836
10010 READ X : POKE I,X : S=S+X : NEXT
10020 DATA 120,169, 51,133,  1,169, 48, 32, 26, 72,169,192
10030 DATA  32, 26, 72,169, 55,133,  1,169, 28,141, 24,208
10040 DATA  88, 96,133,  5,169,208,160,  0,132,  2,132,  4
10050 DATA 133,  3,162, 16,177,  2,145,  4,136,208,249,230
10060 DATA   5,230,  3,202,208,242, 96,173,192,  2,162,193
10070 DATA 160,  2, 32,249,253,169,  2,166,186,160,  0, 32
10080 DATA   0,254,169,  0,162,  0,160,192, 76,213,255,173
10090 DATA 192,  2,162,193,160,  2, 32,249,253,169,  2,166
10100 DATA 186,160,  0, 32,  0,254,169, 20,141, 24,208,169
10110 DATA  48,133,  5,169,192, 32, 30, 72,169,  0,133,  2
10120 DATA 169, 48,133,  3,169,  2,162,255,160, 63, 32,216
10130 DATA 255,120,169, 48,162, 51,134,  1, 32, 26, 72,169
10140 DATA  55,133,  1,169, 28,141, 24,208, 88, 96,160,  0
10150 DATA 169, 32, 32,210,255, 32,210,255,152,  9, 48, 32
10160 DATA 210,255,162,  0, 32,207, 72,169, 29, 32,210,255
10170 DATA 232,224,  8,208,243,169,165, 32,210,255,169, 13
10180 DATA  32,210,255,200,192,  8,208,212, 96,174,254,  2
10190 DATA 172,253,  2,152, 72,138, 72,169,  0, 56,106,202
10200 DATA  16,252, 57,248, 50,208,  3,160,  0, 44,160,  6
10210 DATA 162,  6,185,245, 72, 32,210,255,200,202,208,246
10220 DATA 104,170,104,168, 96,146, 31,111, 31,146,157, 18
10230 DATA  28, 32, 31,146,157,173,252,  2,162,  0,201, 20
10240 DATA 240, 35,201,148,240, 31,232,201, 30,240, 26,201
10250 DATA  94,208,  5,172,251,  2,240, 17,232,172,251,  2
10260 DATA 208, 11,160, 28,232,221, 47, 73,240,  3,136,208
10270 DATA 247,232,142,252,  2, 96, 87, 29, 81,157, 65, 17
10280 DATA  50,145,133,137,134,138,135,139,136,140, 76, 68
10290 DATA  72, 73, 19,  7, 24, 66, 67, 70, 86,169, 36,133
10300 DATA   2,169,  1,162,  2,160,  0, 32,249,253,169,  2
10310 DATA 166,186,160,  0, 32,  0,254,169,  0,162,  0,160
10320 DATA  64,134, 95,132, 96, 32,213,255,165, 95,164, 96
10330 DATA  32, 55,165,173,  0,  3, 72,173,  1,  3, 72,169
```

```
10340 DATA  61,141,  0,  3,169,227,141,  1,  3, 32,195,166
10350 DATA 104,141,  1,  3,104,141,  0,  3, 96
10360 IF S <> 46617 THEN PRINT "FEHLER IN DATAS !!" : END
10370 PRINT "OK" :RETURN
```

```
                  0010         ;
                  0020         ;MASCHINENROUTINEN:
                  0030         ;******************
                  0040         ;
                  0050         ;
                  0060         .OS
                  0070         .BA $4800        ;STARTADRESSE
                  0080         .MC $0800
                  0090         ;
                  0100         ;SPRUNGADRESSEN UND REGISTER:
                  0110         ;****************************
                  0120         ;
                  0130 CHROUT .DE $FFD2        ;ZEICHENAUSGABE
                  0140 FNPAR  .DE $FDF9        ;FILENAMENPARAMETER SETZEN
                  0150 FPAR   .DE $FE00        ;FILEPARAMETER SETZEN
                  0160 SAVE   .DE $FFD8        ;SPEICHERN AUF DISK/KASSETTE
                  0170 LOAD   .DE $FFD5        ;LADEN VON DISK/KASSETTE
                  0180 TESTCH .DE $32F8        ;TESTZEICHEN (ANGEZEIGTES)
                  0190 LAENGE .DE $02C0        ;FILENAMENLAENGE
                  0200 MODUS  .DE $02FB        ;MODUS
                  0210 TASTE  .DE $02FC        ;BEFEHLSTASTENDRUCK
                  0220 YKOORD .DE $02FD        ;FELD-Y-KOORDINATE
                  0230 XKOORD .DE $02FE        ;FELD-X-KOORDINATE
                  0240         ;
                  0250         ;ZEICHENSATZ COPIEREN:
                  0260         ;*********************
                  0270         ;
4800- 78          0280 INIT    SEI              ;KEIN INTERRUPT
4801- A9 33       0290         LDA #$33
4803- 85 01       0300         STA *$01         ;ZEICHENSATZ LESBAR MACHEN
4805- A9 30       0310         LDA #$30         ;VON $D000 NACH $3000
4807- 20 1A 48    0320         JSR MOVE         ;COPIEREN
480A- A9 C0       0330         LDA #$C0         ;VON $D000 NACH $C000
480C- 20 1A 48    0340         JSR MOVE         ;COPIEREN
480F- A9 37       0350         LDA #$37
4811- 85 01       0360         STA *$01         ;I/O AUSWAEHLEN
4813- A9 1C       0370         LDA #$1C
4815- 8D 18 D0    0380         STA $D018        ;ZEICHENSATZADRESSE NACH $3000
4818- 58          0390         CLI              ;INTERRUPT ERMOEGLICHEN
4819- 60          0400         RTS
                  0410         ;
                  0420         ;COPIEREN:
                  0430         ;*********
                  0440         ;
481A- 85 05       0450 MOVE    STA *$05         ;ZIELADRESSE HIGHBYTE
481C- A9 D0       0460         LDA #$D0         ;QUELLADRESSE HIGHBYTE
481E- A0 00       0470 M0      LDY #$00
4820- 84 02       0480         STY *$02         ;QUELLADRESSE LOWBYTE
4822- 84 04       0490         STY *$04         ;ZIELADRESSE LOWBYTE
4824- 85 03       0500         STA *$03
4826- A2 10       0510         LDX #$10
4828- B1 02       0520 M1      LDA ($02),Y      ;LADEN
482A- 91 04       0530         STA ($04),Y      ;SPEICHERN
482C- 88          0540         DEY
482D- D0 F9       0550         BNE M1
482F- E6 05       0560         INC *$05
4831- E6 03       0570         INC *$03
4833- CA          0580         DEX
4834- D0 F2       0590         BNE M1           ;NAECHSTE SEITE
4836- 60          0600         RTS
                  0610         ;
                  0620         ;ZEICHENSATZ LADEN:
```

```
                        0630        ;*****************
                        0640        ;
4837-  AD C0 02         0650 LADEN  LDA LAENGE       ;NAMENLAENGE
483A-  A2 C1            0660        LDX #$C1         ;FILENAMENADRESSE LOW-
483C-  A0 02            0670        LDY #$02         ;HIGHBYTE
483E-  20 F9 FD         0680        JSR FNPAR
4841-  A9 02            0690        LDA #$02         ;LOGISCHE FILENUMMER
4843-  A6 BA            0700        LDX *$BA         ;GERAETEADESSE
4845-  A0 00            0710        LDY #$00         ;SECUNDAERADRESSE
4847-  20 00 FE         0720        JSR FPAR
484A-  A9 00            0730        LDA #$00         ;LOAD/VERIFY-FLAG
484C-  A2 00            0740        LDX #$00         ;STARTADRESSE (LOWBYTE)
484E-  A0 C0            0750        LDY #$C0         ;STARTADRESSE (HIGHBYTE)
4850-  4C D5 FF         0760        JMP LOAD
                        0770        ;
                        0780        ;ZEICHENSATZ SPEICHERN:
                        0790        ;*********************
                        0800        ;
4853-  AD C0 02         0810 SPEICH LDA LAENGE       ;FILENAMENLAENGE
4856-  A2 C1            0820        LDX #$C1         ;FILENAMENADRESSE (LOW)
4858-  A0 02            0830        LDY #$02         ;FILENAMENADRESSE (HIGH)
485A-  20 F9 FD         0840        JSR FNPAR
485D-  A9 02            0850        LDA #$02         ;LOGISCHE FILENUMMER
485F-  A6 BA            0860        LDX *$BA         ;GERAETEADRESSE
4861-  A0 00            0870        LDY #$00         ;SECUNDAERADRESSE
4863-  20 00 FE         0880        JSR FPAR
4866-  A9 14            0890        LDA #$14
4868-  8D 18 D0         0900        STA $D018        ;ORIGINALE ZEICHENSATZLAGE
486B-  A9 30            0910        LDA #$30         ;ZIELADRESSE (HIGHBYTE)
486D-  85 05            0920        STA *$05
486F-  A9 C0            0930        LDA #$C0         ;QUELLADRESSE (HIGHBYTE)
4871-  20 1E 48         0940        JSR M0           ;VON $C000 NACH $3000
4874-  A9 00            0950        LDA #$00
4876-  85 02            0960        STA *$02         ;STARTADRESSE (LOWBYTE)
4878-  A9 30            0970        LDA #$30
487A-  85 03            0980        STA *$03         ;STARTADRESSE (HIGHBYTE)
487C-  A9 02            0990        LDA #$02         ;NULLSEITENADRESSE DER STARTADRESS
487E-  A2 FF            1000        LDX #$FF         ;ENDADRESSE (LOWBYTE)
4880-  A0 3F            1010        LDY #$3F         ;ENDADRESSE (HIGHBYTE)
4882-  20 D8 FF         1020        JSR SAVE         ;SPEICHERE ZS AB $3000
4885-  78               1030        SEI              ;INTERRUPT BLOCKIEREN
4886-  A9 30            1040        LDA #$30         ;ZIELADRESSE (HIGHBYTE)
4888-  A2 33            1050        LDX #$33
488A-  86 01            1060        STX *$01         ;ZEICHENSATZ LESBAR MACHEN
488C-  20 1A 48         1070        JSR MOVE         ;VON $D000 NACH $3000
488F-  A9 37            1080        LDA #$37
4891-  85 01            1090        STA *$01         ;I/O AUSWAEHLEN
4893-  A9 1C            1100        LDA #$1C
4895-  8D 18 D0         1110        STA $D018        ;ZEICHENSATZADRESSE NACH $3000
4898-  58               1120        CLI              ;INTERRUPT ERMOEGLICHEN
4899-  60               1130        RTS              ;ZURUECK NACH BASIC
                        1140        ;
                        1150        ;ARBEITSFELD HERSTELLEN:
                        1160        ;**********************
                        1170        ;
489A-  A0 00            1180 NETZ   LDY #$00         ;ZEILENZAEHLER = 0
489C-  A9 20            1190 N0     LDA #$20         ;"  "
489E-  20 D2 FF         1200        JSR CHROUT
48A1-  20 D2 FF         1210        JSR CHROUT       ;2 LEERZEICHEN
48A4-  98               1220        TYA
48A5-  09 30            1230        ORA #$30         ;ZEILENZAEHLER IN ZIFFER WANDELN
48A7-  20 D2 FF         1240        JSR CHROUT       ;AUSGEBEN
48AA-  A2 00            1250        LDX #$00
48AC-  20 CF 48         1260 N1     JSR PUNKT        ;EINEN PUNKT ZEICHNEN
48AF-  A9 1D            1270        LDA #$1D         ;CURSOR RECHTS
48B1-  20 D2 FF         1280        JSR CHROUT
```

```
48B4- E8          1290          INX              ;NAECHSTER PUNKT
48B5- E0 08       1300          CPX #$08         ;ACHT PUNKTE PRO ZEILE
48B7- D0 F3       1310          BNE N1
48B9- A9 A5       1320          LDA #$A5         ;STRICH (CHR$(165))
48BB- 20 D2 FF    1330          JSR CHROUT
48BE- A9 0D       1340          LDA #$0D         ;CARRIGE RETURN
48C0- 20 D2 FF    1350          JSR CHROUT
48C3- C8          1360          INY              ;NAECHSTE ZEILE
48C4- C0 08       1370          CPY #$08         ;8 ZEILEN
48C6- D0 D4       1380          BNE N0
48C8- 60          1390          RTS
                  1400          ;
                  1410          ;EINE KOORDINATE ZEICHNEN:
                  1420          ;************************
                  1430          ;
48C9- AE FE 02    1440 PUNKT2   LDX XKOORD       ;ANSTEUERUNG DURCH BASIC
48CC- AC FD 02    1450          LDY YKOORD       ;X/Y-KOORDINATE
48CF- 98          1460 PUNKT    TYA
48D0- 48          1470          PHA
48D1- 8A          1480          TXA
48D2- 48          1490          PHA              ;KOORDINATEN RETTEN
48D3- A9 00       1500          LDA #$00
48D5- 38          1510          SEC              ;EIN BIT SETZEN
48D6- 6A          1520 P0       ROR A            ;RICHTIGES BIT HERAUSSUCHEN
48D7- CA          1530          DEX
48D8- 10 FC       1540          BPL P0
48DA- 39 F8 32    1550          AND TESTCH,Y     ;ANDERE BITS DES TESTZEICHENS LOES
48DD- D0 03       1560          BNE P1           ;BIT (=PUNKT) GESETZT?
48DF- A0 00       1570          LDY #$00         ;NEIN
48E1- 2C          1580          .BY $2C          ;BIT-BEFEHL
48E2- A0 06       1590 P1       LDY #$06         ;BIT GESETZT!
48E4- A2 06       1600          LDX #$06
48E6- B9 F5 48    1610 P2       LDA PKTTAB,Y     ;ZEICHEN AUS PUNKTDARSTELLUNGSTABE
48E9- 20 D2 FF    1620          JSR CHROUT
48EC- C8          1630          INY
48ED- CA          1640          DEX
48EE- D0 F6       1650          BNE P2           ;NAECHSTES ZEICHEN
48F0- 68          1660          PLA
48F1- AA          1670          TAX
48F2- 68          1680          PLA
48F3- A8          1690          TAY              ;KOORDINATEN WIEDERHOLEN
48F4- 60          1700          RTS
                  1710          ;
                  1720          ;ZEICHEN FUER EINE KOORDINATE:
                  1730          ;*****************************
                  1740          ;
48F5- 92 1F 6F    1750 PKTTAB   .BY 146 031 111 031 146 157
48F8- 1F 92 9D
48FB- 12 1C 20    1760          .BY 018 028 032 031 146 157
48FE- 1F 92 9D
                  1770          ;
                  1780          ;TASTE ALS BEFEHL IDENTIFIZIEREN:
                  1790          ;********************************
                  1800          ;
4901- AD FC 02    1810 BEFIDE   LDA TASTE        ;TASTENCODE
4904- A2 00       1820          LDX #$00         ;BEFEHLSCODE
4906- C9 14       1830          CMP #$14         ;DEL (=MODUSWECHSEL)?
4908- F0 23       1840          BEQ B2           ;JA
490A- C9 94       1850          CMP #$94         ;INSERT (WIE DEL)?
490C- F0 1F       1860          BEQ B2           ;JA
490E- E8          1870          INX              ;BEFEHLSCODE + 1
490F- C9 1E       1880          CMP #$1E         ;CTRL. "PFEIL HOCH" ?
4911- F0 1A       1890          BEQ B2           ;JA
4913- C9 5E       1900          CMP #$5E         ;"PFEIL HOCH" ?
4915- D0 05       1910          BNE B0           ;NEIN
4917- AC FB 02    1920          LDY MODUS        ;NUR IN MODUS 0
```

```
491A- F0 11     1930         BEQ B2
491C- E8        1940 B0      INX              ;BEFEHLSCODE + 1
491D- AC FB 02  1950         LDY MODUS        ;MODUS 1?
4920- D0 0B     1960         BNE B2           ;JA, DANN KEINE WEITEREN BEFEHLE
4922- A0 1C     1970         LDY #$1C         ;27-1 BEFEHLE (ZAEHLER)
4924- E8        1980 B1      INX              ;BEFEHLSCODE ERHOEHEN
4925- DD 2F 49  1990         CMP BEFTAB-3,X   ;TASTE MIT TABELLE VERGLEICHEN
4928- F0 03     2000         BEQ B2           ;GEFUNDEN
492A- 88        2010         DEY              ;NAECHSTER BEFEHL
492B- D0 F7     2020         BNE B1           ;NOCH NICHT FERTIG
492D- E8        2030 B2      INX
492E- 8E FC 02  2040         STX TASTE        ;BEFEHLSCODE ALS RUECKMELDUNG
4931- 60        2050         RTS              ;ZURUECK ZU BASIC
                2060         :
                2070         ;BEFEHLSTASTENTABELLE:
                2080         ;*********************
                2090         :
                2100         :    W/CRSR-RE/Q/CRSR-LI/A/CRSR-UN
4932- 57 1D 51  2110 BEFTAB .BY 087 029  081  157  065  017
4935- 9D 41 11
                2120         :    2/CRSR-OB/ F1/ F2/ F3/ F4
4938- 32 91 85  2130         .BY 050 145   133 137 134 138
493B- 89 86 8A
                2140         ;    F5/ F6/ F7/ F8/ L / D
493E- 87 8B 88  2150         .BY 135 139 136 140 076 068
4941- 8C 4C 44
                2160         ;    H / I/<HOME>/CTRL.G/CTRL.X/ B
4944- 48 49 13  2170         .BY 072 073 019   007     024  066
4947- 07 18 42
                2180         ;    C / F / V
494A- 43 46 56  2190         .BY 067 070 086
                2200         :
                2210         ;DISKCATALOG:
                2220         ;************
                2230         ;
494D- A9 24     2240 CATALG LDA #$24          ;"$" ALS FILENAME
494F- 85 02     2250         STA *$02
4951- A9 01     2260         LDA #$01
4953- A2 02     2270         LDX #$02
4955- A0 00     2280         LDY #$00         ;S.O.
4957- 20 F9 FD  2290         JSR FNPAR
495A- A9 02     2300         LDA #$02
495C- A6 BA     2310         LDX *$BA         ;GERAETEADRESSE
495E- A0 00     2320         LDY #$00
4960- 20 00 FE  2330         JSR FPAR
4963- A9 00     2340         LDA #$00         ;LOAD/VERIFY-FLAG
4965- A2 00     2350         LDX #$00
4967- A0 40     2360         LDY #$40         ;STARTADRESSE
4969- 86 5F     2370         STX *$5F
496B- 84 60     2380         STY *$60
496D- 20 D5 FF  2390         JSR LOAD         ;LADE CATALOG WIE BASICPROGRAMM
4970- A5 5F     2400         LDA *$5F         ;SIMULIERE:
4972- A4 60     2410         LDY *$60         ;BASIC-PROGRAMMSTARTADRESSE
4974- 20 37 A5  2420         JSR $A537        ;BASICZEILEN BINDEN
4977- AD 00 03  2430         LDA $0300
497A- 48        2440         PHA
497B- AD 01 03  2450         LDA $0301        ;ORIGINALEN WARMSTARTVEKTOR
497E- 48        2460         PHA              ; RETTEN
497F- A9 3D     2470         LDA #$3D
4981- 8D 00 03  2480         STA $0300
4984- A9 E3     2490         LDA #$E3
4986- 8D 01 03  2500         STA $0301        ;UND AUF RTS SETZEN
4989- 20 C3 A6  2510         JSR $A6C3        ;LISTBEFEHL AUSFUEHREN
498C- 68        2520         PLA
498D- 8D 01 03  2530         STA $0301
4990- 68        2540         PLA
```

```
4991- 8D 00 03  2550        STA $0300         ;ALTEN VEKTOR WIEDERHOLEN
4994- 60        2560        RTS               ;ZURUECK ZU BASIC
                2570        .EN
END OF ASSEMBLY!


 ---  LABEL FILE:  --


B0 =491C                B1 =4924
B2 =492D                BEFIDE =4901
BEFTAB =4932            CATALG =494D
CHROUT =FFD2            FNPAR =FDF9
FPAR =FE00              INIT =4800
LADEN =4837             LAENGE =02C0
LOAD =FFD5              M0 =481E
M1 =4828                MODUS =02FB
MOVE =481A              N0 =489C
N1 =48AC                NETZ =489A
P0 =48D6                P1 =48E2
P2 =48E6                PKTTAB =48F5
PUNKT =48CF             PUNKT2 =48C9
SAVE =FFD8              SPEICH =4853
TASTE =02FC             TESTCH =32F8
XKOORD =02FE            YKOORD =02FD

//0000,4995,0995
```

Der Zeichenformer funktioniert analog zu dem Spriteeditor von Paragraph 5.3. Auch hier existieren ein Sekundäroperations-, ein Farbzuteilungs-, ein Editor- (hier 8x8) und ein Originalfeld. Hinzu kommt noch die sogenannte Modus- und Satzangabe. Ihr Programm kennt zwei Modi:

- 0: Eingabemodus:
  In Modus 0 können Sie alle Befehle benutzen
  und Ihr eigenes Zeichen herstellen.

- 1: Abrufmodus:
  Modus 1 ermöglicht Ihnen einen bequemen Abruf
  bereits hergestellter Zeichen per Tastendruck
  aus dem Zeichenbuffer (Nach dem Starten Ihres
  Programmes enthält der zuständige Zeichenbuffer
  den kompletten originalen Zeichensatz).

Zur Satzangabe sind die 4 Zeichensätze (normal-Groß/Graphik // invers-Groß/Graphik // normal-Klein/Groß // invers-Klein/Groß) mit Ihren jeweils 128 Zeichen von 1-4 durchnumeriert. Der aktuelle Satz, der gerade behandelt wird, steht dann in der Satzangabe.

Grundsätzlich können alle Befehle des Spriteeditor (bis auf G) auch im Zeichenformer angewandt werden und sind dort nachzulesen. Zusätzlich stehen Ihnen hier noch folgende Funktionen zur Verfügung:

- *<del>* -

Moduswechsel (Modus 0-1)

- *<ctrl>^* -

Wollen Sie die Nummer des Zeichensatzes wechseln, so drücken Sie <ctrl>^ und die Nr. (1-4) des Satzes ein (in Modus 0 genügt auch ^).

- D/H -

Mit D können Sie Ihr Zeichen einem ASCII-Zeichen, bzw. einer Taste zuordnen und damit in Ihren Zeichensatz übernehmen.
Mit H holen Sie sich ein bereits existierendes Zeichen aus dem Zeichenbuffer in ihr Feld (analog zu Modus 1).

Bei beiden wird der aktuelle Zeichensatz (s.o.)(1-4) angesprochen. Nun entscheiden Sie, ob Sie das Zeichen als ASCII-Wert (f5/f6) oder Tastendruck (f1/f3) angeben wollen.
Jetzt erfolgt die Eingabe der Taste oder des ASCII-Wertes mit <return>!

- <ctrl>S / <ctrl>G -

Abspeichern oder Laden eines ganzen Zeichensatzes. Näheres siehe Spriteeditor.

Haben Sie sich mit diesem komfortablen Zeicheneditor Ihren eigenen, persönlichen Zeichensatz hergestellt und auf Diskette gespeichert, dann können Sie Ihn nach dem Beenden des Editorprogramms ganz einfach mit LOAD "name",8,1 nach $3000 (12288) einladen (um ihn woanders hin zu laden, brauchen Sie einen Monitor oder ein Maschinensprache-Ladeprogramm). Als Nächstes müssen Sie mit dem folgenden einfachen Befehl den originalen Zeichensatzvektor verschieben, so daß der VIC sich die benötigten Definitionen von Stund an aus diesem Bereich holt:

```
POKE 53248+24, PEEK(53248+24) AND 241 OR 12
```

In diesem Befehl werden die drei Bits 1-3 des 24. VIC-Registers, die die Lage bzw. die Adreßbits 11-13 des Zeichengenerators bestimmen, zunächst gelöscht (241 = %1111 0001) und dann die obersten beiden der drei Bits gesetzt (12 = %0000 1100). Schlagartig erscheint Ihre neue Zeichenkreation auf dem Bildschirm. So einfach ist das.

5.5 Eingabe/Ausgabe von Graphik und Zeichensatz

Mitunter dauert es recht lange, bis wir mühselig ein Graphikbild ersonnen und auf den Bildschirm gebracht haben. Sollten wir dies jedesmal machen, wenn wir uns das Bild betrachten wollen, so ginge uns recht bald die Lust verloren. Doch es gibt einige Möglichkeiten, um diesen Prozess abzukürzen. Zum einen können wir unser Bild auf Diskette oder Kassette abspeichern und bei Bedarf wieder in den Speicher holen. Zum anderen, was das Anschauen später noch weiter verkürzt, sind wir in der Lage - sofern Sie einen Drucker besitzen, der gleichfalls Graphik ausgeben kann (z.B. durch Einzelnadelansteuerung) - sogenannte Hardcopies anzufertigen, also Bilder auf dem Blatt Papier. Die notwendigen Techniken für diese Unterfangen werden Ihnen hier vorgestellt. Es ist klar, daß wir keine Hardcopyroutinen für alle möglichen Drucker angeben können, da leider fast jeder Drucker seine eigene Art und Weise der Graphikausgabe besitzt. Deshalb sollten Sie einmal mit dem hier Gesagten versuchen, sich selbst eine eigene Routine für Ihren Drucker zu schreiben. Notfalls schauen Sie sich einmal in den entsprechenden Fachzeitschriften um, die ab und zu einmal verschiedene Artikel zu diesem Thema bringen. Die Supergraphik bietet darüberhinaus bereits 3 verschiedene Hardcopy-Routinen für die verschiedendsten Drucker.

5.5.1 Abspeichern/Laden

Das Problem bei der Ein- und Ausgabe von Graphiken oder eines Zeichensatzes auf Diskette oder Kassette ist das Abspeichern, da hierbei dem Computer Anfangs- und Endadresse des zu speichernden Bereiches angegeben werden muß und kein entsprechender Befehl hierzu existiert. Bei normalen BASIC-Programmen sind dem Rechner diese Dinge bekannt, Maschinenprogramme oder sonstige Daten jedoch müssen auf etwas kompliziertere Art und Weise auf das Speichermedium übertragen werden.

Grundsätzlich kann dies durch die folgende Routine geschehen. Sie ist wieder so gehalten, daß Sie sich reibungslos in Ihre gesammelten Graphikroutinen einfügt:

```
10 REM ******************************
20 REM **                          **
30 REM **  GRAPHIKABSPEICHERUNG    **
40 REM **                          **
50 REM ******************************
60 REM
70 FI$="GRAPHIK" : GA = 8 : REM FILENAME UND GERAETEADRESSE
80 BE = 8192 : EN = 16192 : REM START- UND ENDADRESSE (HIER GRAPHI K BEI $2000)
90 GOSUB 11200 : END : REM SPEICHERN
11200 REM
11210 REM   *******************
11220 REM   **  ABSPEICHERN  **
11230 REM   *******************
11240 REM
11250 SYS 57812 FI$,GA : REM DISKPARAMETERUEBERNAME (AN $E1D4)
11260 X = EN/256
11270 POKE 175, INT(X) : REM HIGH-BYTE ENDADRESSE ($AF)
11280 POKE 174, (X-INT(X))*256 : REM LOW-BYTE ENDADRESSE ($AE)
11290 X = BE/256
11300 POKE 194, INT(X) : REM HIGH-BYTE STARTADRESSE ($C2)
11310 POKE 193, (X-INT(X))*256 : REM LOW-BYTE STARTADRESSE ($C1)
11320 SYS 62954 : REM SAVE-ROUTINE ($F5EA)
11330 RETURN
```

In diesem Programm werden in den Zeilen 70 und 80 der eigentlichen Speicherroutine die notwendigen Informationen zur Erzeugung eines Files übergeben. Zeile 11250 ruft dann einen Teil des normalen BASIC SAVE-Befehls auf, der Filenamen und Geräteadresse (für Disk: SA=8; für Kassette: SA=1) übernimmt. Ihnen mag dieser eine Befehl etwas ungewohnt erscheinen, diese Form ist aber durchaus korrekt. Nachdem nämlich mit dem SYS 57812 die besagte Routine aufgerufen wurde, werden, wie beim SAVE-Befehl, die beiden Parameter verlangt. Aus diesem Grunde gibt es keinen SYNTAX ERROR.

Die folgenden Zeilen übergeben der in Zeile 11320 aufgerufenen eigentlichen SAVE-Routine in bestimmten Speicherstellen die Start- und die Endadresse des zu speichernden Bereiches.

Wir können dieses Programm für alle fraglichen Funktionen verwenden, die wir im Auge hatten. Sowohl Zeichensätze, wie Sprites, Graphik-, Text- und Farbspeicher können so auf Diskette oder Kassette gesichert werden. Ausschlaggebend ist dabei lediglich die Wahl der verschiedenen Parameter. Oben wurde das Beispiel zur Speicherung von Graphik gegeben, die in dem Bereich von $2000-$3F3F (8192-16192) liegt. Für unsere Zeichensätze, die wir in $3000-

$3FFF (12288-16383) gespeichert hatten, müßten Sie die Zeile 80 wie folgt ändern:

80 BE = 12288 : EN = 16384

Wie Sie sehen, müssen Sie zur Endadresse stets 1 hinzuzählen. Möchten Sie vielleicht einmal Ihre gesamte Textseite auf Diskette bringen, die bekanntlich von $0400 bis $07E7 (1024-2023) geht, so schreiben Sie:

80 BE = 1024 : EN = 2024

Gleiches tippen Sie ein, wenn Sie den Farbteil Ihrer Graphik (sofern er dort liegt) auf Band oder Diskette bringen wollen. Vielleicht aber haben Sie auch eine Spritedefinition in Block 11 und wollen diese abspeichern, z.B. um sie dem Spriteeditor aus Kapitel 5.3 zugänglich zu machen (in diesem Fall muß die Definition in Block 11 liegen) oder einfach, um Sie später wieder direkt hinein zu laden. Da Block 11 bei $02C0-$02FD (704-766) liegt, muß die entsprechende Passage lauten:

80 BE = 704 : EN = 767

Nebenbei können Sie damit auch jedes Maschinenprogramm speichern, ohne einen Monitor zu Rate ziehen zu müssen.

Wollen wir die verschiedenen Dinge wieder einladen, so genügt ein LOAD "name",8,1 (bei Floppy-Besitzern) oder für Kassettenrecorder: LOAD "name",1,1 und schon ist die Sache erledigt.

5.5.2 Hardcopy

Die Archillesferse aller kommerziellen Programme ist der Druckerbetrieb. Da es 'zig verschiedene Druckertypen gibt und natürlich jeder seine eigenen Ansteuerungen, ASCII-Codierung oder Steuerzeichen besitzt (besonders, was die Graphik betrifft), gibt es kaum Programme, die mehr als ein oder höchstens zwei Druckertypen bedienen können. Wenn Sie sich z.B. nach einer Graphikbefehlserweiterung oder einem anderen (Graphik-) Programm umschauen, so gebe ich Ihnen den Tip, unbedingt darauf zu achten, daß dieses Programm auch Ihren Drucker bedienen kann, und falls Sie noch keinen Drucker besitzen, besonderes Augenmerk auf die Programme zu legen, die möglichst viele verschiedene Druckertypen ansteuern können. Denn Sie werden sich, auch wenn Sie zur Zeit vielleicht noch keinen Wert darauf legen,

garantiert nach einiger Zeit irgendeinen Drucker zulegen. Allein schon die Tatsache, daß etwas größere Programme ohne Druckerlisting völlig unüberschaubar werden, zwingt einen nach einiger Zeit zum Drucker. Falls Sie mit Graphik arbeiten, so ist unbedingt auf die Graphikfähigkeit eines solchen Gerätes zu achten (u.a. auch die Sauberkeit des Druckes, mit der ein Bild aufs Papier gebracht wird, die besonders bei Graphikausdrucken ins Auge fällt). Sie werden es sonst noch einmal bereuen!

Wie oben bereits gesagt, ist es hier unmöglich, alle Druckertypen mit den entsprechenden Routinen vorzustellen. Aus diesem Grund werden wir uns hier mit dem wohl gängigsten Gerät, dem Seikosha GP 100 VC beschäftigen, auf dem eine graphische Hardcopy aufgrund des 7-Nadelkopfes wohl am kompliziertesten ist. Routinen für andere Drucker können Sie sich dabei gut davon ableiten. Angenommen wird wieder eine Graphikseite bei $2000-$3F3F (8192-16191).

```
11800 REM *****************************
11810 REM **        GRAPHIK-         **
11820 REM **  HARDCOPY - GP 100 VC   **
11830 REM **                         **
11840 REM *****************************
11850 REM
11860 SA = 8192
11870 OPEN1,4 : REM DRUCKERKANAL OEFFNEN
11880 Y=35:MK=255:ZZ=28:SG=6:REM Y-KOORD. / MASKE / ZEILENZAHL / S
PALTENGROESSE
11890 FOR FLAG=1 TO 2
11900  FOR ZE=1 TO ZZ : REM ZZ ZEILEN
11910   PRINT#1,CHR$(8)CHR$(27)CHR$(16)CHR$(0)CHR$(80);:REM MITTENZENTRIERT+GRA
PHIK EIN
11920   XK=0
11930   FOR SP=1 TO 40 : YK=Y : REM SPALTENZAEHLER
11940    FOR X=0 TO SG:GOSUB 12150:ZW(X)=PEEK(AD):YK=YK+1:NEXT X:REM 8*7 PUNKTE
 LADEN
11950    FOR X=0 TO 7 : REM 8 BYTES ZUM DRUCKER
11960     MS=2^(7-X):B2=0
11970     FOR Z=0 TO SG:B1=-2*((ZW(Z) AND MS)>0):B2=B2 OR (B1^Z+(Z +B1=0)):NEXT
Z
11980     B2=(B2 AND MK) OR 128:PRINT#1,CHR$(B2);
11990    NEXT X : REM NAECHSTES DRUCKERBYTE
12000    XK=XK+8
12010   NEXT SP : REM NAECHSTE SPALTE
12020   PRINT#1 : REM RETURN
12030   Y=Y+7 : REM Y-KOORD.
12040  NEXT ZE : REM NAECHSTE ZEILE
12050  ZZ=1 : REM NUR LETZTE ZEILE
```

```
12060  MK = 16 : REM MASKE (OBERE 4 BITS ABSCHNEIDEN)
12070  SG=4 : REM SPALTENGROESSE
12080 NEXT FLAG : NOCH EINMAL FUER DIE LETZTE ZEILE
12090 CLOSE 1:END
12100 REM
12110 REM ************************
12120 REM **  PUNKTBERECHNUNG  **
12130 REM ************************
12140 REM
12150 AD = SA + 320 * INT(YK/8) + (YK AND 7) + 8 * INT(XK/8) : RETURN
```

Auch diese Routine können Sie wieder in Ihren Unterprogrammschatz aufnehmen. Sie brauchen dann nur noch per GOSUB aufgerufen werden. Sie können jedoch auch Ihre Graphik bei SA=8192 ($2000) erstellen, dann dieses Programm hier einladen und laufen lassen. Da dabei die Graphik nicht zerstört wurde, erscheint eine originalgetreue Abbildung des Graphikspeichers auf dem Druckerpapier. Diese Routine funktioniert natürlich auch auf allen kompatiblen Druckern, wie beispielsweise Epson mit DATA-BECKER-Interface. Doch so eine Routine in BASIC dauert sehr lange. Geben Sie also nicht gleich den Mut auf, wenn Sie etwas warten müssen. Schließlich gibt es ja noch die Supergraphik mit ihren diversen Drucker-Routinen in Assembler. Sicher kann dieses Programm, wie auch alle anderen "frisiert" werden (wie das zu machen ist, wird im Anhang geschildert), doch hier wurde der Übersichtlichkeit wegen geordnet vorgegangen. Wichtig zum Verständnis dieser Routine ist die Tatsache, daß stets 7 untereinander liegende Punkte in einem Byte an den Drucker übergeben werden (jedes gesetzte Bit resultiert in einem gesetzten Punkt auf dem Papier), wobei das High-Bit stets gesetzt sein muß (dies erfordert der Drucker). Dadurch taucht eine kleine Schwierigkeit am Graphikende auf: Es bleiben nämlich 4 Reihen übrig, da 200 (die Zahl der Punkte in y-Richtung) nicht durch 7 teilbar ist. Diese werden in einem weiteren Durchlauf mit ein paar veränderten Parameter übersandt. In dem Hauptteil der Routine (Zeilen 11940-11990) werden zunächst 7 untereinander liegende Graphikbytes eingelesen (Z. 11940), die dann Spalte für Spalte an den Drucker gegeben werden (Z. 11950-11990). Die weiteren Einzelheiten zu besprechen, würde den Rahmen sprengen. Nur eine Kleinigkeit sollte hier noch erwähnt werden: Die Operation Z+B1=0, wie Sie in Zeile 11970 durchgeführt wird, ergibt -1, wenn Z+B1=0 ist und 0, wenn dies nicht der Fall ist. Aus 4=6 z.B. resultiert 0, aus 4=4 jedoch -1. Probieren Sie es aus. Dieser Trick kann eine wertvolle Hilfe in vielen Anwendungen sein, in denen Entscheidungen gefordert werden, die Sie ungern mit IF oder ON...GOTO tätigen (z.B. aus Zeitgründen). In dem Fall hier wird damit die Tatsache umgangen, daß der Commodore 64 aus 0^0 1 herausbekommt,

was falsche Resultate bringen würde. In dem Graphik-Aid weiter unten und im Supergraphik-Listing wird diese Hardcopy-Routine in Assembler angegeben.

5.6 IRQ-Handhabung

Aus Abschnitt 4.7 kennen Sie bereits die verschiedenen Interrupt-(Unterbrechungs-) Möglichkeiten, die bei graphischen Anwendungen interessant sind. Dort wurde Ihnen diese Fähigkeit vorgestellt und alle theoretischen Grundlagen geliefert. Jetzt, da wir auch einigen praktischen Unterbau in Sachen Graphik besitzen, ist es an der Zeit, das wohl schwierigste aller Kapitel zu beginnen: Die Interruptprogrammierung. Schon die Tatsache, daß Interrupts nur von Maschinensprache aus zu bedienen sind, macht das ganze für den Uneingeweihten nebulös und undurchsichtig. Aber auch erfahrene Assemblerprogrammierer müssen sich mit den völlig neuen Programmiertechniken vertraut machen. Doch haben Sie keine Angst. Sie können die angegebenen Beispiele ruhig ausprobieren. Sie werden Ihnen besonders hübsche Effekte zeitigen.

Die Interrupttechnik soll Ihnen anhand des Rasterzeilen-IRQs und des Lightpens gezeigt werden. Danach können Sie die erfahrenen Kenntnisse auf Ihre eigenen Programme mit z.B. Sprite-Kollisionen etc. anwenden. Bevor Sie sich jedoch an diesen Abschnitt begeben, sollten Sie zunächst Paragraph 4.7 gut durchgelesen haben.

Erinnern wir uns kurz an die wesentlichen Faktoren, die zur Bedienung von allgemeinen IRQs des VIC wissensnotwendig sind:

Interrupts sind kontrollierte Unterbrechungen des Programmablaufs. Es gibt 4 verschiedene Ursachen, durch die der Videocontroller Interrupts auslösen kann: Rasterzeilen, Lightpen, Sprite-Sprite-, und Sprite-Hintergrundzeichen-Kollisionen. Diese sind durch Setzen der entsprechenden Bits im Interrupt Mask Register (IMR), also VIC-Register 26, auszuwählen. Wird ein Interrupt ausgelöst, so erfolgt die Rückmeldung durch Setzen des korrespondierenden Bits in Register 25 des VIC, das sogenannte Interrupt Request Register (IRR); das 7. Bit dieser Speicherstelle wird dabei als Kennzeichen ebenfalls gesetzt. Dieses Register ist nach jedem Interrupt durch Rückschreiben seines Inhalts zu löschen. Bei jedem IRQ wird eine sogenannte Interruptroutine aufgerufen, deren Adresse in den Speicherstellen $0314/$0315 (788/789) steht. Ein IRQ kann in Maschinensprache durch das Setzen des Interruptflags unterbunden werden.

5.6.1 Rasterzeilen-IRQ

Auch hier sollten wir uns zunächst an die notwendigen Dinge erinnern, die für Rasterzeilen-Interrupts nützlich sind:

Für diese Art von Unterbrechung sind jeweils die Bits 0 der IRR und IMR zuständig. Aus den Registern 18 und 17/Bit 7 erfahren Sie die aktuelle Bildschirmzeile, die der VIC gerade aufbaut. Das vom VIC verwandte Koordinatensystem unterscheidet sich von dem normalen, uns bekannten (s. Kap. 4.7). Werden diese beiden Register beschrieben, so geben Sie damit an, bei welcher Rasterzeile ein Interrupt stattfinden soll.

Mit diesen Informationen ist es uns jetzt möglich, eine eigene Interruptroutine zu erzeugen und damit direkt in das Geschehen einzugreifen. Dazu sei diese als Assemblerlisting angegeben:

```
100: C800                     *=   $C800
110: C800           FARBE1  =    $FB
120: C800           FARBE2  =    $FC
130: C800           OBEN    =    $FD
140: C800           UNTEN   =    $FE
150: C800           IRQVECT =    $0314
160: C800           IRQALT  =    $EA31
170: C800           RASTER  =    $D012
180: C800           IRR     =    $D019
190: C800           IMR     =    $D01A
200: C800           RAHMEN  =    $D020
210: C800           HGRUND  =    $D021
220:                 ;
230: C800            ;INITIALISIEREN:
240:                 ;****************
250:                 ;
260: C800 78        INIT    SEI          ;INTERRUPT VERHINDERN
270: C801 A9 1F             LDA #< IRQNEU
280: C803 8D 14 03          STA IRQVECT
290: C806 A9 C8             LDA #> IRQNEU
300: C808 8D 15 03          STA IRQVECT+1 ;IRQ-VEKTOR UMLEGEN
310: C80B A5 FD             LDA OBEN
320: C80D 8D 12 D0          STA RASTER   ;1.RASTERZ. FESTLEGEN
330: C810 AD 11 D0          LDA RASTER-1
340: C813 29 7F             AND #$7F
350: C815 8D 11 D0          STA RASTER-1  ;HIGH-BIT LOESCHEN
360: C818 A9 81             LDA #%10000001  ;MASKE
370: C81A 8D 1A D0          STA IMR      ;RASTERZ.-IRQ WAEHLEN
380: C81D 58                CLI          ;IRQ ERMOEGLICHEN
```

```
390: C81E 60              RTS
400:                ;
410: C81F           ;INTERRUPTROUTINE:
420:                ;******************
430:                ;
440: C81F AD 19 D0 IRQNEU  LDA IRR     ;INTERRUPTREGISTER
450: C822 8D 19 D0          STA IRR     ;LOESCHEN
460: C825 30 07             BMI IRQRAS  ;RASTERZ.-IRQPRINT
470:                ;NORMALER IRQ:
480: C827 AD 0D DC          LDA $DC0D   ;CIA 1-IRR LOESCHEN
490: C82A 58                CLI         ;INTERR. ERMOEGLICHEN
500: C82B 4C 31 EA          JMP IRQALT  ;ZUR ALTEN ROUTINE
510: C82E AD 12 D0 IRQRAS  LDA RASTER  ;RASTERPOSITION
520: C831 C5 FE             CMP UNTEN   ;UNTERER WERTPRINT
530: C833 B0 10             BCS ZWEITER  ;JA=>SPRUNG
540: C835 A5 FB             LDA FARBE1
550: C837 8D 20 D0          STA RAHMEN  ;RAHMEN- UND
560: C83A 8D 21 D0          STA HGRUND  ;HINTEGRUNDFARBE
570: C83D A5 FE             LDA UNTEN   ;UNTEREN WERT IN
580: C83F 8D 12 D0          STA RASTER  ;RASTERPOSITION
590: C842 4C BC FE          JMP $FEBC   ;ABSCHLIESSEN
600: C845 A5 FC   ZWEITER  LDA FARBE2
610: C847 8D 20 D0          STA RAHMEN  ;RAHMEN- UND
620: C84A 8D 21 D0          STA HGRUND  ;HINTERGRUNDFARBE
630: C84D A5 FD             LDA OBEN    ;OBEREN WERT IN
640: C84F 8D 12 D0          STA RASTER  ;RASTERPOSITION
650: C852 4C BC FE          JMP $FEBC   ;ABSCHLIESSEN
```

Und hier das entsprechende BASIC-Ladeprogramm mit einer kleinen hübschen Anwendung:

```
1000 FOR I = 51200 TO 51284
1010 READ X : POKE I,X : S=S+X : NEXT
1020 DATA 120,169, 31,141, 20,  3,169,200,141, 21,  3,165
1030 DATA 253,141, 18,208,173, 17,208, 41,127,141, 17,208
1040 DATA 169,129,141, 26,208, 88, 96,173, 25,208,141, 25
1050 DATA 208, 48,  7,173, 13,220, 88, 76, 49,234,173, 18
1060 DATA 208,197,254,176, 16,165,251,141, 32,208,141, 33
1070 DATA 208,165,254,141, 18,208, 76,188,254,165,252,141
1080 DATA  32,208,141, 33,208,165,253,141, 18,208, 76,188
1090 DATA 254
1100 IF S <> 11288 THEN PRINT "FEHLER IN DATAS !!" : END
1110 PRINT "OK"
1120 F1 = 7 : F2 = 6 : REM FARBEN 1 UND 2 (STREIFEN IN FARBE 1)
1130 OB = 60 : UN = 150 : REM OBERE UND UNTERE KANTE DES STREIFENS
1140 POKE 251,F1:POKE 252,F2:POKE 253,OB:POKE 254,UN : REM UEBERGEBEN
1150 SYS 51200 : REM ROUTINE EINSCHALTEN
1160 REM
```

```
1170 FOR X=1 TO 5000 : NEXT X : REM WARTESCHLEIFE
1180 REM
1190 REM BEWEGEN:
1200 REM ********
1210 FOR X=40 TO 240 : POKE 253,X : POKE 254, X+10 : NEXT X
1220 GET A$ : IF A$="" THEN 1190 : REM TASTE?
```

Wie Sie sich vielleicht erinnern, wird auch in der Supergraphik reger Gebrauch von den verschiedensten Möglichkeiten der Interrupanwendung gemacht. Die Interrupt-Routinen der Supergraphik beginnen im Listing (groß hervorgehoben) beim Label IRQNEU und beinhalten die Routinen zum Bewegen der Sprites, zur Graphik-Text-Mischanzeige (mittels Rasterinterrupt) und zur Tonsteuerung. Besonders interessant, aber ebenso kompliziert ist die Spritebewegung, bei der Gebrauch von der HLINE-Routine zum Berechnen von Linien gemacht wird. Da nun gleichzeitig diese Routine einmal vom Hauptprogramm und von der IRQ-Routine verwendet werden kann, muß die IRQ-Routine vor dem Ansprung von HLINE sämtliche dort verwendeten Register retten. Vielleicht schauen Sie sich das einmal an.

5.6.2 Lightpen

Bitte lesen Sie ruhig weiter, auch wenn Sie keinen Lightpen besitzen, es springt auch etwas für Sie ab!

Was ein Lightpen oder Lichtgriffel ist, wie er funktioniert und in den Rechnerablauf integriert wird, wissen Sie bereits aus dem Abschnitt 4.7.2. Fassen wir hier das wichtigste noch einmal kurz zusammen:

Der Lightpen ist ein Instrument, mit dem der Computer beliebige Positionen des Bildschirms identifizieren kann, auf die der Stift gerade zeigt. Die jeweiligen x- und y-Koordinaten dieses Punktes können durch Lesen der VIC-Register 19 und 20 festgestellt werden. Das Koordinatenraster entspricht dem der Rasterzeilen (s. Kap. 4.7). Auch der Lightpen kann einen Interrupt auslösen. In den IMR und IRR sind dafür jeweils die 3. Bits zuständig. Sein Eingang am Controlport 1 stimmt mit dem Eingang für den Feuerknopf eines Joysticks überein.

Doch wie ist ein Lightpen anzuwenden, wie wird er programmiert? Die Lightpenabfrage kann auf zwei verschiedene Arten durchgeführt werden. Die einfachere ist die durch ein kleines BASIC-Programm, das lediglich die beiden Register ausliest, die die Bildschirmkoordinaten enthalten. Ein Beispiel wäre etwa das Zeichnen eines Punktes an

die Stelle, auf die der Lightpen gerichtet ist. Diese Funktion übernimmt das folgende kleine Demonstrationsprogramm, das nur zusammen mit den Graphikroutinen aus dem Abschnitt 5.2 funktionstüchtig ist!

```
100 REM ****************
110 REM **            **
120 REM **  LIGHTPEN  **
130 REM **            **
140 REM ****************
150 REM
160 V = 53248 : REM VIC BASISADRESSE
170 SA = 8192 : REM GRAPHIKSTARTADRESSE
180 GOSUB 10000:GOSUB 10200:FA=7*16+2:GOSUB 10400 : REM GRAPHIK INITIALI
SIEREN
190 XP = PEEK(V+19) : REM LIGHTPEN-X-KOORDINATE
200 YP = PEEK(V+20) : REM LIGHTPEN-Y-KOORDINATE
210 XK = 2 * (XP - 40) : REM ERRECHNE X-KOORD.
220 YK = YP - 40 : REM ERRECHNE Y-KOORD.
230 IF XK>319 OR XK<0 OR YK>199 OR YK<0 THEN 190 : REM AUF BEREICH TESTE
N
240 GOSUB 10700 : REM PUNKT ZEICHNEN
250 GOTO 190
10000 REM INITIALISIERUNGS+PUNKTSETZ-ROUTINEN
10010 REM ......
10020 REM ...      (S. KAPITEL 5.2)
```

Wenn Sie sich hier den Umstand mit den BASIC-Routinen ersparen wollen, laden Sie die Supergraphik und ersetzen die folgenden Zeilen:

```
180 GMODE 0,1 : GCLEAR : SCOL=0,2 : SCOL=1,7
240 PLOT ,XK,YK
```

Die Zeilen 160 und 170, sowie ab 10000 wären dann überflüssig.

In diesem kleinen Programm werden - nach dem Einschalten und Löschen der Graphik - in den Zeilen 190 und 200 lediglich die x- und y-Koordinaten des Punktes auf dem Bildschirm eingelesen, bei dem der Lightpen zuletzt auflag. Diese Bildschirmkoordinaten werden dann in unsere bekannten Graphikkoordinaten nach der Formel umgerechnet, die in Abschnitt 4.7.2 angegeben wurde (Z. 210/220). Alsdann muß das Ergebnis darauf geprüft werden, ob der Punkt, den der Lightpen anzeigte nicht außerhalb des Bildschimfensters liegt, was ne-

gative oder zu große Werte resultieren ließ. Erst dann kann die Punkt-Setz-Routine aufgerufen werden, um an der jeweiligen Stelle eben einen Punkt zu setzen. Sie können dieses Programm beliebig erweitern mit vielen Funktionen, so daß evt. ein ganzes Zeichenprogramm entsteht. Vielleicht legen Sie mit dem Lightpen die Eckpunkte von Linien oder den Mittelpunkt eines Kreises und seinen Radius fest, die auf Tastendruck dann gezeichnet werden, oder wählen per Lightpen bestimmte Menüfunktionen aus, die am Bildrand angezeigt werden.

Doch eine Sache stört dabei doch. Egal ob der Lightpen auf den Bildschirm zeigt oder nicht, an dieser Stelle wird ein Punkt gezeichnet. Hier mag das noch nicht so schlimme Folgen haben, bei anderen Anwendungen wird es sicher stören. Eine andere Möglichkeit der Bedienung ist die per Interrupt. Jedesmal, wenn der Lightpen einen Impuls sendet, also nur dann, wenn er auch tatsächlich auf den Bildschirm zeigt, könnte beispielsweise ein IRQ ausgelöst werden, der entweder einen Punkt zeichnet oder einfach in einer Speicherstelle ein kurzes Signal gibt, das nach einiger Zeit wieder gelöscht wird.

Eine andere Möglichkeit ist beispielsweise die Konstruktion einer Alarmanlage: Der Lightpen wird an die Zimmerlampe montiert, oder einfach auf sie gerichtet. Sobald irgendjemand diese Lampe beim Eintreten einschaltet, sendet der Lightpen einen Impuls an den Computer, der sofort mit einem Heulgetöse beginnt, das sämtliche Nachbarn aus dem Bett wirft (man könnte den Audioausgang vielleicht an eine Stereoanlage anschließen (geht mit einem einfachen Überspielkabel), die dem Ganzen noch etwas mehr "Power" gibt). Eine Variation dieser Alarmanlage wäre es z.B., wenn das Aufgehen der Türe einen Kontakt auslöste, der eine Lampe zum Leuchten brächte, was seinerseits wieder über den Lightpen zu einer Reaktion des Computers führte.

Wir wollen uns statt der Auslösung eines Alarms mit einem Farbwechsel des Rahmens zufrieden geben:

```
100: C800                 *=   $C800
110: C800           FARBE  =   $FB
120: C800           IRQVECT=   $0314
130: C800           IRQALT =   $EA31
140: C800           IRR    =   $D019
150: C800           IMR    =   $D01A
160: C800           RAHMEN =   $D020
170:                ;
180:                ;INITIALISIEREN:
190:                ;****************
200:                ;
210: C800 78        INIT   SEI          ;INTERRUPT VERHINDERN
220: C801 A9 16            LDA #< IRQNEU
230: C803 8D 14 03         STA IRQVECT
240: C806 A9 C8            LDA #> IRQNEU
250: C808 8D 15 03         STA IRQVECT+1 ;IRQ-VEKTOR UMLEGEN
260: C80B A9 00            LDA #$00
270: C80D 85 FB            STA FARBE    ;FARBE SCHWARZ
280: C80F A9 88            LDA #%10001000 ;MASKE
290: C811 8D 1A D0         STA IMR      ;LIGHTPEN-IRQ WAEHLEN
300: C814 58               CLI          ;IRQ ERMOEGLICHEN
310: C815 60               RTS
320:                ;
330:                ;INTERRUPTROUTINE:
340:                ;******************
350:                ;
360: C816 AD 19 D0 IRQNEU LDA IRR      ;INTERRUPTREGISTER
370: C819 8D 19 D0        STA IRR      ;LOESCHEN
380: C81C 30 07           BMI IRQRAS   ;RASTERZ.-IRQPRINT
390:                ;NORMALER IRQ
400: C81E AD 0D DC        LDA $0C0D    ;CIA 1-IRR LOESCHEN
410: C821 58              CLI          ;INTERRUPT ERMOEGLICHEN
420: C822 4C 31 EA        JMP IRQALT   ;ZUR ALTEN ROUTINE
430: C825 A5 FB    IRQRAS LDA FARBE
440: C827 8D 20 D0        STA RAHMEN   ;RAHMENFARBE
450: C82A E6 FB           INC FARBE    ;FARBE ERHOEHEN
460: C82C 4C BC FE        JMP $FEBC    ;ABSCHLIESSEN
```

Und hier das Ladeprogramm:

```
100 FOR I = 51200 TO 51246
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,169, 22,141, 20,  3,169,200,141, 21,  3,169
130 DATA   0,133,251,169,136,141, 26,208, 88, 96,173, 25
140 DATA 208,141, 25,208, 48,  7,173, 13,220, 88, 76, 49
150 DATA 234,165,251,141, 32,208,230,251, 76,188,254
160 IF S <> 5910 THEN PRINT "FEHLER IN DATAS !!" : END
170 PRINT "OK"
```

Sie starten wie immer mit einem

```
SYS 51200
```

Alle im folgenden genannten Zeilennummern beziehen sich auf das Assemblerlisting.

Die Initialisierungsroutine, die in Zeile 220 beginnt, sollte Ihnen inzwischen schon geläufig sein. Auch hier wird erst wieder der IRQ-Vektor umgelegt auf unsere eigene Routine und der entsprechende Interrupt diesmal durch Setzen des 3. Bits des IMR (Interrupt Mask Register) eingeschaltet. Nichts Besonderes bietet gleichfalls der Einstieg in unsere Interruptroutine (ab Zeile 360). Wieder wird auf die Art des Interrupts geprüft und entsprechend verzweigt. Dann aber kommt der Wechsel der Rahmenfarbe. Der Vorgang an sich ist leicht zu verstehen, wichtig hierbei ist allerdings, daß zum erstenmal eine reguläre Speicherstelle durch die Interruptroutine verändert wird. Dies bringt besondere Effekte, ist aber auch besonders gefährlich, wenn man nicht genau weiß, was mit dieser Stelle wann passiert, denn der Interrupt kann ja grundsätzlich zu jeder Zeit und in jeder Routine ausgelöst werden. Sie können aber von außen, also beispielsweise von BASIC aus, den momentanen Zustand der IRQ-Routine abfragen oder andere Steuerfunktionen übernehmen. Dies wird z.B. vom Betriebssystem genutzt, indem die interne Unterbrechungsroutine jedesmal die TI$-Uhr verstellt, die dann von BASIC aus abgefragt werden kann. Auch hier natürlich stehen Ihnen viele Möglichkeiten offen.

Bitte erinnern Sie sich, daß sie dieses Bildschirmblinken auch durch Betätigen des Feuerknopfes Ihres in Port 1 gesteckten Joysticks erzeugen können.

Sie sehen, der Interrupt ist ein schönes "Spielzeug", das enorme Welten entstehen läßt, wenn man es richtig zu nutzen weiß. Beschäftigen Sie sich ruhig ein wenig damit, es lohnt sich!

5.7 Ein kleines Graphik-Paket

Wir haben eine ganze Menge über Graphik und Ihre Programmierung gehört und gelesen. Viele Routinen sollten uns das Leben erleichtern und ein wenig Einblick in die phantastische Welt der Bilder vermitteln. Doch beim Ausprobieren all dieser Utilities und Basic-Programme, die wir gespannt eingetippt hatten, wurden wir zwar nicht durch das Ergebnis enttäuscht, dafür aber umso mehr durch den Weg dorthin. Bärte und graue Haare waren keine Seltenheit: Unsere Basicprogramme waren so langsam, auch wenn wir die gut gemeinten Tips im Anhang beherzigten, daß da kaum was von den Wogen der graphischen Dramatik hinüberschwappte. Kurz: Wir wissen zwar, wie es geht, doch die richtige Befriedigung machte sich kaum bemerkbar. Dem soll in diesem Abschnitt abgeholfen werden. Hier wird erst einmal anständig ausgepackt. Ein relativ umfangreiches Graphikpaket in Maschinensprache macht Ihre Graphik schnell und interessant.

Dieses Graphikpaket soll vor allem dazu dienen, Ihnen die Realisierung der einzelnen Figuren in Maschinensprache zu demonstrieren. Sie können anhand der hier vorgestellten, umfangreich dokumentierten Routinen erkennen, wie Sie auf einfache Weise Maschinenprogramme von BASIC aus aufrufen und diverse Parameter übergeben können.

Weiterhin soll Ihnen dieses Graphikpaket die Möglichkeit geben, einzelne Graphikroutinen leicht in eigene Programme einzubauen, ohne den großen Aufbau der Supergraphik mitnehmen zu müssen.

Wenn Sie sich die nächsten Seiten betrachten, sollten Sie nicht kapitulieren. Es ist zwar viel Arbeit, die ganzen Maschinenroutinen abzuschreiben und auf Fehler zu kontrollieren, doch es lohnt sich! Wer wirklich ernsthaft mit Graphik arbeiten will, der sollte vor diesem Zeit- und Mußeaufwand nicht zurückschrecken, der braucht einfach das entsprechende Handwerkszeug.

Doch nun zum Thema: Zunächst wird Ihnen das Assemblerlisting des Graphik-Paketes mit vielen Kommentaren vorgestellt, für diejenigen, die sich für die einzelnen Routinen interessieren. Sie werden sehen, daß einige Dinge aus Geschwindigkeitsgründen etwas anders organisiert sind, als wir es in den obigen Basicprogrammen getan haben. Dazu gehört z.B. die Routine, die beliebige Linien auf den Bildschirm zeichnet. Die theoretischen Grundlagen zu dieser Linienzeichnung in Assembler finden Sie ebenfalls in dem Abschnitt zum Zeichnen von Linien.

Hier noch einmal in Kürze:
In dieser Routine werden nicht direkt die Koordinaten eines jeden Punktes ausgerechnet, aus denen ja dann wieder die Speicheradresse errechnet werden müßte, sondern wir berechnen quasi das Verhältnis der Anzahl der zu gehenden Schritte nach oben/unten zu der Anzahl von Schritten nach rechts/links, um vom Ausgangspunkt zum Endpunkt zu gelangen, und gehen dann stets in einem bestimmten Rythmus, der diesem Verhältnis entspricht, eben in diese Richtungen. Dabei wird die Tatsache voll berücksichtigt, daß stets zwei Punkte über- bzw. nebeneinander liegen müssen, um eine zusammenhängende Linie zu erhalten. Sie werden staunen, wie schnell so etwas geht.

Nach diesem sogenannten Source-Listing wird Ihnen wieder ein Basic-Lader für diejenigen, die keinen Monitor besitzen, angeboten, um das Programm sicher und gut eintippen zu können. Doch hier das Listing:

```
C800                 *=   $C800
           ;
           ;
           ; **************************
           ; **                      **
           ; **  HOCH AUFLOESENDES   **
           ; **   GRAPHIK - PAKET    **
           ; **                      **
           ; **************************
           ;
           ;
           ;
           ;ROM-SPRUNGADRESSEN:
           ;********************
           ;
B79E       GETBYT   =   $B79E   ;HOLT BYTEWERT
AEFD       CHKCOM   =   $AEFD   ;PRUEFT AUF KOMMA
B7F1       CHKGET   =   $B7F1   ;CHKCOM+GETBYT
B7EB       GETCOR   =   $B7EB   ;HOLT KOORDINATEN
B248       QERR     =   $B248   ;ILLEGAL QUANTITY
D000       V        =   $D000   ;VIDEOCONTROLLER
           ;
```

```
              ;NULLSEITIGE REGISTER:
              ;**********************
              ;
0063          OFFX     =    $63
00AB          MSK      =    $AB      ;MASKE
0069          DIF0     =    $69
006A          DIF1     =    $6A
006B          DIF2     =    $6B
006C          DIF3     =    $6C
006D          DIF4     =    $6D
006E          DIF5     =    $6E
006F          ZWN      =    $6F      ;XK/8 (HLINE)
0070          ZA       =    $70      ;ZAEHLER (HLINE)
00AC          A        =    $AC      ;PUNKTADRESSE
00AD          B        =    $AC+1
0014          XK       =    $14      ;X-KOORDINATE
0097          FLG      =    $97      ;UNPLOT/PLOT-FLAG
00FD          USE      =    $FD      ;DIVERSES
              ;
              ;FESTE WERTE:
              ;*************
              ;
2000          GRAPH    =    $2000    ;GRAPHIKSTART
0400          VIDEO    =    $0400    ;VIDEORAMSTART
0021          GSTART   =    $21      ;GR-START+1-H-BYTE
003E          GEND     =    $3E      ;GR-ENDE-1-H-BYTE
              ;
              ;
              ;ANSTEUERADRESSEN:
              ;*****************
              ;
              ;
C800 4C 24 C8           JMP  INIT    ;GRAPHIK EIN
C803 4C 41 C8           JMP  GOFF    ;GRAPHIK AUS
C806 4C 12 C9           JMP  GCLEAR  ;GRAPHIK LOESCHEN
C809 4C 31 C9           JMP  SCOLOR  ;FARBE SETZEN (LOESCHEN)
C80C 4C 2A C9           JMP  PCOLOR  ;PLOTFARBE
C80F 4C 58 C8           JMP  PLOT    ;PUNKT SETZEN
C812 4C 55 C8           JMP  UNPLOT  ;PUNKT LOESCHEN
C815 4C 6B C8           JMP  SLINE   ;LINIE ZEICHNEN
C818 4C 68 C8           JMP  CLLINE  ;LINIE LOESCHEN
C81B 4C 43 CA           JMP  GLOAD   ;GRAPHIK LADEN
C81E 4C 52 CA           JMP  GSAVE   ;GRAPHIK SPEICHERN
C821 4C 69 CA           JMP  HARDC   ;HARDCOPY (GP 100 VC ETC.)
              ;
              ;
```

```
              ;GRAPHIK EINSCHALTEN:
              ;*********************
              ;
C824 EA       INIT     NOP            ;KEINE BLOCKADE
C825 AD 11 D0          LDA  V+17
C828 8D 1E CB          STA  STORE1
C82B AD 18 D0          LDA  V+24
C82E 8D 1F CB          STA  STORE2    ;ALTE INHALTE RETTEN
C831 A9 3B             LDA  #%00111011
C833 8D 11 D0          STA  V+17      ;GRAPHIK EIN
C836 A9 18             LDA  #%00011000
C838 8D 18 D0          STA  V+24      ;NACH $2000
C83B A9 60             LDA  #$60      ;CODE FUER RTS ALS BLOCKADE
C83D 8D 24 C8          STA  INIT      ;INIT WIRD BLOCKIERT
C840 60                RTS
              ;
              ;
              ;GRAPHIK AUSSCHALTEN:
              ;*********************
              ;
C841 AD 1E CB GOFF     LDA  STORE1
C844 8D 11 D0          STA  V+17
C847 AD 1F CB          LDA  STORE2
C84A 8D 18 D0          STA  V+24      ;ALTE INHALTE RUECKHOLEN
C84D A9 EA             LDA  #$EA      ;CODE FUER NOP FUER
C84F 8D 24 C8          STA  INIT      ;BLOCKADE AUFHEBEN
C852 4C 44 E5          JMP  $E544     ;BILDSCHIRM LOESCHEN
              ;
              ;
              ;PUNKT LOESCHEN:
              ;***************
              ;
C855 A2 00    UNPLOT   LDX  #$00      ;LOESCH-FLAG
C857 2C                .BYT $2C
              ;
              ;
              ;PUNKT SETZEN:
              ;*************
              ;
C858 A2 80    PLOT     LDX  #$80      ;SETZ-FLAG
C85A 86 97    PL1      STX  FLG
C85C 20 FD AE          JSR  CHKCOM
C85F 20 79 C8          JSR  TESCOR    ;KOORDINATEN HOLEN
C862 20 94 C8          JSR  HPOSN     ;ADRESSE ERRECHNEN
C865 4C E2 C8          JMP  PLT       ;PUNKT SETZEN/LOESCHEN
              ;
              ;
```

```
              ;LINIE LOESCHEN:
              ;****************
              ;
C868 A2 00    CLLINE   LDX  #$00     ;LOESCH-FLAG
C86A 2C                .BYT $2C
              ;
              ;
              ;LINIE ZEICHNEN:
              ;****************
              ;
C86B A2 80    SLINE    LDX  #$80     ;SETZ-FLAG
C86D 20 5A C8          JSR  PL1      ;ERSTEN PUNKT SETZEN
C870 20 FD AE          JSR  CHKCOM
C873 20 79 C8          JSR  TESCOR   ;ZWEITE KOORD. HOLEN
C876 4C BB C9          JMP  HLINE    ;LINIE ZEICHNEN
              ;
              ;
              ;KOORDINATEN TESTEN:
              ;********************
              ;
C879 20 EB B7 TESCOR   JSR  GETCOR   ;HOLEN
C87C 8A                TXA
C87D A8                TAY
C87E A6 15             LDX  XK+1
C880 C0 C8             CPY  #200     ;Y-KOORD. >= 200?
C882 B0 0D             BCS  ILLFF    ;JA!
C884 A5 14             LDA  XK
C886 E0 01             CPX  #>320    ;X-KOORD. >= 320?
C888 90 06             BCC  T1       ;JA
C88A D0 05             BNE  ILLFF    ;A":XK-LOW
C88C C9 40             CMP  #<320    ;X":XK-HIGH
C88E B0 01             BCS  ILLFF    ;Y":YK
C890 60       T1       RTS
C891 4C 48 B2 ILLFF    JMP  QERR     ;ILLEGAL QUANTITY
              ;
              ;
              ;ADRESSE ERRECHNEN:
              ;******************
              ;
C894 8C 1C CB HPOSN    STY  YK       ;Y-K
C897 8D 1A CB          STA  XKL      ;X-KL
C89A 8E 1B CB          STX  XKH      ;X-KH (ZWISCHENSPEICHERN)
C89D 85 14             STA  XK
C89F 86 15             STX  XK+1
C8A1 98                TYA
C8A2 4A                LSR  A
C8A3 4A                LSR  A
C8A4 4A                LSR  A        ;INT(Y/8)
C8A5 AA                TAX
C8A6 BD 2F CB          LDA  MUL.H,X ;320*INT(Y/8) (HIGH-BYTE)
C8A9 85 AD             STA  B
```

```
C8AB 8A                 TXA
C8AC 29 03              AND  #3       ;BITS 0+1 ISOLIEREN
C8AE AA                 TAX
C8AF BD 49 CB           LDA  MUL.L,X ;320*INT(Y/8) (LOW-BYTE)
C8B2 85 AC              STA  A
C8B4 98                 TYA           ;Y-KOORD.
C8B5 29 07              AND  #7       ;(Y AND 7)
C8B7 18                 CLC
C8B8 65 AC              ADC  A        ;OFFY=320*INT(Y/8)+(Y AND 7)
C8BA 85 AC              STA  A        ;*****************************
C8BC A5 14              LDA  XK
C8BE 29 F8              AND  #$F8     ;OFFX=8*INT(X/8)
C8C0 85 63              STA  OFFX     ;****************
C8C2 A9 20              LDA  #>GRAPH ;GRAPHIKSEITE
C8C4 05 AD              ORA  B
C8C6 85 AD              STA  B        ;+SA
C8C8 18                 CLC
C8C9 A5 AC              LDA  A
C8CB 65 63              ADC  OFFX     ;AD = OFFY + OFFX + SA
C8CD 85 AC              STA  A        ;**********************
C8CF A5 AD              LDA  B
C8D1 65 15              ADC  XK+1
C8D3 85 AD              STA  B
C8D5 A5 14              LDA  XK
C8D7 29 07              AND  #7
C8D9 49 07              EOR  #7         ;7-(X AND 7)
C8DB AA                 TAX
C8DC BD 4D CB           LDA  MSKTAB,X  ;MASKENTABELLE
C8DF 85 AB              STA  MSK        ;2^(7-(X AND 7))
C8E1 60                 RTS
               ;
               ;
               ;PUNKT PLOTTEN:
               ;***************
               ;
C8E2 A0 00     PLT      LDY  #0
C8E4 08                 PHP           ;CARRY-FLAG RETTEN
C8E5 A5 AB              LDA  MSK      ;MASKE
C8E7 24 97              BIT  FLG      ;PLOT/UNPLOT-FLAG
C8E9 30 05              BMI  PL2      ;PLOT
C8EB 49 FF              EOR  #$FF     ;UNPLOT
C8ED 31 AC              AND  (A),Y
C8EF 2C                 .BYT $2C      ;UEBERSPRINGE NAECHSTEN BEFEHL
C8F0 11 AC     PL2      ORA  (A),Y    ;PLOT
C8F2 91 AC              STA  (A),Y
C8F4 A5 AC              LDA  A        ;FARBE SETZEN
C8F6 85 FD              STA  USE
C8F8 A5 AD              LDA  B
C8FA 4A                 LSR  A
C8FB 66 FD              ROR  USE
C8FD 4A                 LSR  A
```

```
C8FE 66 FD              ROR  USE
C900 4A                 LSR  A
C901 66 FD              ROR  USE      ;ADRESSE/8
C903 29 03              AND  #$03
C905 09 04              ORA  #$04     ;VIDEORAM AB $0400
C907 85 FE              STA  USE+1
C909 AD 1D CB           LDA  COLOR    ;FARBE IN
C90C 91 FD              STA  (USE),Y  ;VIDEORAM
C90E 28                 PLP           ;CARRY-FLAG
C90F A4 6F              LDY  ZWN
C911 60                 RTS
               ;
               ;
               ;GRAPHIK LOESCHEN:
               ;*****************
               ;
C912 A9 20     GCLEAR   LDA  #>GRAPH
C914 85 FE              STA  USE+1
C916 A0 00              LDY  #<GRAPH   ;Y=0
C918 84 FD              STY  USE      ;GRAPHIK-STARTADRESSE
C91A A2 20              LDX  #$20     ;LAENGE
C91C 98                 TYA           ;Y=0!
C91D 91 FD     GC1      STA  (USE),Y  ;LOESCHEN
C91F C8                 INY
C920 D0 FB              BNE  GC1
C922 E6 FE              INC  USE+1
C924 CA                 DEX
C925 D0 F6              BNE  GC1
C927 4C 37 C9           JMP  SCOL1    ;VIDEORAM LOESCHEN
               ;
               ;
               ;PUNKTFARBE DEFINIEREN:
               ;**********************
               ;
C92A 20 F1 B7  PCOLOR   JSR  CHKGET   ;KOMMA+FARBE
C92D 8E 1D CB           STX  COLOR
C930 60                 RTS
               ;
               ;
               ;FARBE SETZEN:
               ;*************
               ;
C931 20 F1 B7  SCOLOR   JSR  CHKGET   ;KOMMA+FARBE
C934 8E 1D CB           STX  COLOR
C937 A2 03     SCOL1    LDX  #3
C939 A9 04              LDA  #>VIDEO
C93B 85 FE              STA  USE+1
C93D A0 00              LDY  #<VIDEO
C93F 84 FD              STY  USE      ;ADRESSE VIDEORAM
C941 84 97              STY  FLG      ;Y=0!
C943 AD 1D CB           LDA  COLOR    ;FARBE
```

```
C946 91 FD    GM9     STA  (USE),Y ;SETZEN
C948 C8               INY
C949 C4 97            CPY  FLG
C94B D0 F9            BNE  GM9
C94D E6 FE            INC  USE+1
C94F CA               DEX
C950 F0 03            BEQ  GM9.
C952 10 F2            BPL  GM9
C954 60               RTS
C955 A2 E8    GM9.    LDX  #$E8
C957 86 97            STX  FLG      ;SPRITEVEKTOREN SCHUETZEN
C959 D0 EB            BNE  GM9      ;UNBEDINGT
              ;
              ;
              ;VEKTORROUTINEN:
              ;****************
              ;
C95B A5 AC    UNTEN   LDA  A        ;PUNKTADRESSE-LOW
C95D 29 07            AND  #7
C95F C9 07            CMP  #7
C961 F0 05            BEQ  UN1      ;PACKENRAND!
C963 38               SEC
C964 A9 00            LDA  #0
C966 B0 04            BCS  UN2      ;ADDIERE 1 (C=1!)
C968 A9 38    UN1     LDA  #$38     ;ADDIERE 320-7=313
C96A E6 AD            INC  B        ;C=1!
C96C 65 AC    UN2     ADC  A
C96E 85 AC            STA  A
C970 A9 00            LDA  #0
C972 65 AD            ADC  B
C974 85 AD            STA  B
C976 60               RTS
              ;
C977 30 E2    U.O     BMI  UNTEN
              ;
C979 A5 AC    OBEN    LDA  A        ;ADRESSE
C97B 29 07            AND  #7
C97D F0 05            BEQ  OB1      ;PACKENRAND (OBEN)
C97F 18               CLC
C980 A9 FF            LDA  #$FF
C982 90 04            BCC  OB2      ;SUBTRAHIERE 1
C984 A9 C7    OB1     LDA  #$C7     ;SUBTAHIERE 320+7=327
C986 C6 AD            DEC  B        ;C=1!
C988 65 AC    OB2     ADC  A
C98A 85 AC            STA  A
C98C A5 AD            LDA  B
C98E E9 00            SBC  #0
C990 85 AD            STA  B
C992 60               RTS
              ;
              ;
```

```
C993 46 AB     RECHTS  LSR  MSK      ;MASKE VERSCHIEBEN
C995 90 0E             BCC  RE2      ;NOCH INNERHALB BYTE
C997 66 AB             ROR  MSK      ;AUSSERHALB BYTE
C999 A5 AC             LDA  A
C99B C8                INY
C99C 18                CLC
C99D 69 08             ADC  #8
C99F 85 AC             STA  A
C9A1 90 02             BCC  RE2
C9A3 E6 AD             INC  B        ;8 ADDIEREN
C9A5 60        RE2     RTS
               ;
C9A6 10 EB     R.L     BPL  RECHTS
               ;
C9A8 06 AB     LINKS   ASL  MSK
C9AA 90 0E             BCC  LI1
C9AC 26 AB             ROL  MSK
C9AE A5 AC             LDA  A
C9B0 88                DEY
C9B1 38                SEC
C9B2 E9 08     LI3     SBC  #8
C9B4 85 AC             STA  A
C9B6 B0 02             BCS  LI1
C9B8 C6 AD             DEC  B        ;8 SUBTRAHIEREN
C9BA 60        LI1     RTS
               ;
               ;
               ;LINIE ZEICHNEN:
               ;****************
               ;
C9BB 48        HLINE   PHA           ;A ": X2-LOW-BYTE
C9BC AD 1B CB          LDA  XKH      ;X ": X2-HIGH-BYTE
C9BF 4A                LSR  A        ;Y ": Y2
C9C0 AD 1A CB          LDA  XKL      ;XKL": X1-LOW-BYTE
C9C3 6A                ROR  A        ;XKH": X1-LOW-BYTE
C9C4 4A                LSR  A        ;YK": Y1
C9C5 4A                LSR  A
C9C6 85 6F             STA  ZWN      ;X1 / 8 ZWISCHENSP.
C9C8 68                PLA
C9C9 48                PHA
C9CA 38                SEC
C9CB ED 1A CB          SBC  XKL
C9CE 48                PHA
C9CF 8A                TXA
C9D0 ED 1B CB          SBC  XKH
C9D3 85 6C             STA  DIF3     ;X2-X1
C9D5 B0 0A             BCS  L3       ;NEGATIV?
C9D7 68                PLA           ;JA
C9D8 49 FF             EOR  #$FF
C9DA 69 01             ADC  #1
C9DC 48                PHA
```

```
C9DD A9 00              LDA  #0
C9DF E5 6C              SBC  DIF3     ;VORZEICHENWECHSEL
C9E1 85 6A      L3      STA  DIF1
C9E3 85 6E              STA  DIF5     ;(X2-X1)-HIGH
C9E5 68                 PLA
C9E6 85 69              STA  DIF0
C9E8 85 6D              STA  DIF4     ;(X2-X1)-LOW
C9EA 68                 PLA
C9EB 8D 1A CB           STA  XKL
C9EE 8E 1B CB           STX  XKH
C9F1 98                 TYA
C9F2 18                 CLC
C9F3 ED 1C CB           SBC  YK       ;Y2-Y1
C9F6 90 04              BCC  L4       ;NEGATIV?
C9F8 49 FF              EOR  #$FF
C9FA 69 FE              ADC  #$FE     ;VORZEICHENWECHSEL
C9FC 85 6B      L4      STA  DIF2     ;(Y2-Y1)
C9FE 8C 1C CB           STY  YK
CA01 66 6C              ROR  DIF3     ;(X2-X1)/2
CA03 38                 SEC
CA04 E5 69              SBC  DIF0     ;(Y2-Y1)-(X2-X1)
CA06 AA                 TAX           ;LOW-BYTE NACH X-REG.
CA07 A9 FF              LDA  #$FF
CA09 E5 6A              SBC  DIF1
CA0B 85 70              STA  ZA       ;HIGH-BYTE NACH ZA
CA0D A4 6F              LDY  ZWN
CA0F B0 05              BCS  L5       ;UNBEDINGT
CA11 0A         L1      ASL  A
CA12 20 A6 C9           JSR  R.L      ;RECHTS/LINKS
CA15 38                 SEC
CA16 A5 6D      L5      LDA  DIF4
CA18 65 6B              ADC  DIF2
CA1A 85 6D              STA  DIF4
CA1C A5 6E              LDA  DIF5
CA1E E9 00              SBC  #0       ;(X2-X1)-(Y2-Y1)NACH(X2-X1)
CA20 85 6E      L2      STA  DIF5
CA22 84 6F              STY  ZWN
CA24 20 E2 C8           JSR  PLT      ;PUNKT ZEICHNEN
CA27 E8                 INX
CA28 D0 05              BNE  L6
CA2A E6 70              INC  ZA
CA2C D0 01              BNE  L6       ;DEC ZAEHLER
CA2E 60                 RTS
CA2F A5 6C      L6      LDA  DIF3
CA31 B0 DE              BCS  L1
CA33 20 77 C9           JSR  U.O      ;UNTEN/OBEN
CA36 18                 CLC
CA37 A5 6D              LDA  DIF4
CA39 65 69              ADC  DIF0
CA3B 85 6D              STA  DIF4
CA3D A5 6E              LDA  DIF5
```

```
CA3F 65 6A              ADC  DIF1
CA41 50 DD              BVC  L2      ;UNBEDINGT
               ;
               ;
               ;GRAPHIK LADEN:
               ;***************
               ;
CA43 20 FD AE GLOAD     JSR  CHKCOM  ;KOMMA?
CA46 20 D4 E1           JSR  $E1D4   ;PARAMETER HOLEN
CA49 A0 20              LDY  #>GRAPH
CA4B A2 00              LDX  #<GRAPH ;STARTADRESSE
CA4D A9 00              LDA  #$00    ;LOAD-FLAG
CA4F 4C D5 FF           JMP  $FFD5   ;LADEN
               ;
               ;
               ;GRAPHIK SPEICHERN:
               ;*******************
CA52                                 ;
CA52 20 FD AE GSAVE     JSR  CHKCOM  ;KOMMA?
CA55 20 D4 E1           JSR  $E1D4
               ;ENDADRESSE
CA58 A2 3F              LDX  #>GRAPH+8000
CA5A A0 40              LDY  #<GRAPH+8000
CA5C A9 00              LDA  #<GRAPH
CA5E 85 FD              STA  $FD
CA60 A9 20              LDA  #>GRAPH
CA62 85 FE              STA  $FE     ;STARTADRESSE
CA64 A9 FD              LDA  #$FD    ;POINTER
CA66 4C D8 FF           JMP  $FFD8   ;SPEICHERN
               ;
               ;
               ;HARDCOPY SEIKOSHA GP 100 VC:
               ;*****************************
               ;
CA69 20 F1 B7 HARDC     JSR  CHKGET  ;HOLE LOG. FILENR.
CA6C 86 67              STX  $67
CA6E 20 0F F3           JSR  $F30F   ;SUCHT LOG. FILENR.
CA71 20 1F F3           JSR  $F31F   ;SETZT FILEPARAM.
CA74 A6 67              LDX  $67
CA76 20 C9 FF           JSR  $FFC9   ;KANAL OEFFNEN
CA79 A9 FF              LDA  #$FF
CA7B 85 61              STA  $61     ;MASKE
CA7D A9 07              LDA  #7
CA7F 85 FD              STA  USE     ;PACKENGROESSE
CA81 A9 1C              LDA  #28
CA83 85 97              STA  FLG     ;ZEILENZAEHLER
CA85 A9 00              LDA  #0
CA87 8D 20 CB           STA  ZWIS    ;YK-MERKER
CA8A A9 28    HA1       LDA  #40
CA8C 8D 21 CB           STA  FLG2    ;PACKENZAEHLER
CA8F A2 04              LDX  #4
```

```
CA91 BD 2A CB HA1.     LDA  HATAB,X ;MITTENZENTRIERT
CA94 20 D2 FF          JSR  $FFD2   ;AUSGABE
CA97 CA                DEX
CA98 10 F7             BPL  HA1.
CA9A A9 00             LDA  #0
CA9C 85 63             STA  $63
CA9E 85 64             STA  $64     ;XK=0
CAA0 AD 20 CB HA2      LDA  ZWIS
CAA3 85 65             STA  $65     ;YK
CAA5 A9 00             LDA  #0
CAA7 85 FE             STA  USE+1   ;BUFFERZEIGER
CAA9 A5 63    HA3      LDA  $63     ;XK-L
CAAB A6 64             LDX  $64     ;XK-H
CAAD A4 65             LDY  $65     ;YK
CAAF 20 94 C8          JSR  HPOSN   ;POSITION BERECHNEN
CAB2 A0 00             LDY  #0
CAB4 B1 AC             LDA  (A),Y   ;BYTE HOLEN
CAB6 A6 FE             LDX  USE+1   ;BUFFERZEIGER
CAB8 9D 22 CB          STA  BUFF,X  ;IN BUFFER
CABB E6 65             INC  $65     ;YK
CABD E8                INX
CABE 86 FE             STX  USE+1
CAC0 E4 FD             CPX  USE
CAC2 D0 E5             BNE  HA3
CAC4 A9 00             LDA  #0
CAC6 A0 07             LDY  #7
CAC8 A6 FD    HA4      LDX  USE
CACA 1E 22 CB HA5      ASL  BUFF,X  ;EIN BIT HOLEN
CACD 2A                ROL  A       ;UND IN AKU SCHIEBEN
CACE CA                DEX          ;BUFFERZEIGER ERHOEHEN
CACF 10 F9             BPL  HA5
CAD1 25 61             AND  $61     ;BEI LETZTER ZEILE=#$0F
CAD3 09 80             ORA  #$80    ;HIGH-BYTE
CAD5 20 D2 FF          JSR  $FFD2   ;SENDEN
CAD8 88                DEY          ;8 BYTES SENDEN
CAD9 10 ED             BPL  HA4
CADB A5 63             LDA  $63     ;XK-L
CADD 18                CLC
CADE 69 08             ADC  #8
CAE0 85 63             STA  $63     ;XK+8
CAE2 90 02             BCC  HA6
CAE4 E6 64             INC  $64     ;XK-H
CAE6 CE 21 CB HA6      DEC  FLG2
CAE9 D0 B5             BNE  HA2
CAEB A9 0D             LDA  #$0D    ;CARRIGE RETURN
CAED 20 D2 FF          JSR  $FFD2   ;SENDEN
CAF0 AD 20 CB          LDA  ZWIS
CAF3 18                CLC
CAF4 69 07             ADC  #7
CAF6 8D 20 CB          STA  ZWIS    ;YK+7
CAF9 C6 97             DEC  FLG
```

```
CAFB F0 03              BEQ  HA8.
CAFD 4C 8A CA HA8       JMP  HA1
CB00 A9 04    HA8.      LDA  #4
CB02 C5 FD              CMP  USE
CB04 F0 0C              BEQ  HA7
CB06 85 FD              STA  USE      ;LETZTE ZEILE
CB08 A9 01              LDA  #1
CB0A 85 97              STA  FLG      ;FLAG
CB0C A9 0F              LDA  #$F
CB0E 85 61              STA  $61      ;MASKE
CB10 D0 EB              BNE  HA8
CB12 A9 0F    HA7       LDA  #15      ;NORMAL MODUS
CB14 20 D2 FF           JSR  $FFD2
CB17 4C CC FF           JMP  $FFCC    ;SCHLIESST KANAL
              ;
              ;
              ;INTERNE SPEICHER:
              ;******************
              ;
CB1A 00       XKL       .BYT 0        ;XK-LOW-ZWISCHENSPEICHER
CB1B 00       XKH       .BYT 0        ;XK-HIGH-ZWISCHENSPEICHER
CB1C 00       YK        .BYT 0        ;YK-ZWISCHENSPEICHER
CB1D 00       COLOR     .BYT 0        ;FARBE
CB1E 00       STORE1    .BYT 0        ;V+17-REG.-ZWISCHENSP.
CB1F 00       STORE2    .BYT 0        ;V+24-REG.-ZWISCHENSP.
CB20 00       ZWIS      .BYT 0        ;ZWISCHENSPEICHER
CB21 01       FLG2      .BYT 1        ;ZWISCHENSPEICHER
CB22 00 00 00 BUFF      .WOR 0,0,0,0  ;BUFFER FUER HARDCOPY
              ;
              ;
              ;TABELLEN:
              ;*********
              ;
              ;DRUCKERZEICHEN:
              ;(RUECKWAERTS)
              ;
              ;MITTENZENTRIERT/GRAPHIK EIN
              ;
CB2A 50 00 10 HATAB     .BYT 80,0,16,27,8
              ;
              ;
              ;MULTIPLIKATIONSTABELLE:
              ;(N*320 FUER N=0 BIS N=24)
              ;
              ;HIGH-BYTES
              ;
CB2F 00 01 02 MUL.H     .BYT 0,1,2,3,5,6,7,8,10,11,12,13,15,16
CB3D 11 12 14           .BYT 17,18,20,21,22,23,25,26,27,28,30,31
              ;
```

```
               ;LOW-BYTES
               ;
CB49 00 40 80 MUL.L     .BYT $00,$40,$80,$C0
               ;
               ;
               ;MASKENTABELLE:
               ;
CB4D 01 02 04 MSKTAB    .BYT %00000001,%00000010,%00000100,%00001000
CB51 10 20 40           .BYT %00010000,%00100000,%01000000,%10000000
```

Symboltabelle:

A	00AC	B	00AD	BUFF	CB22
CHKCOM	AEFD	CHKGET	B7F1	CLLINE	C868
COLOR	CB1D	DIF0	0069	DIF1	006A
DIF2	006B	DIF3	006C	DIF4	006D
DIF5	006E	FLG	0097	FLG2	CB21
GC1	C91D	GCLEAR	C912	GEND	003E
GETBYT	B79E	GETCOR	B7EB	GLOAD	CA43
GM9	C946	GM9.	C955	GOFF	C841
GRAPH	2000	GSAVE	CA52	GSTART	0021
HA1	CA8A	HA1.	CA91	HA2	CAA0
HA3	CAA9	HA4	CAC8	HA5	CACA
HA6	CAE6	HA7	CB12	HA8	CAFD
HA8.	CB00	HARDC	CA69	HATAB	CB2A
HLINE	C9BB	HPOSN	C894	ILLFF	C891
INIT	C824	L1	CA11	L2	CA20
L3	C9E1	L4	C9FC	L5	CA16
L6	CA2F	LI1	C9BA	LI3	C9B2
LINKS	C9A8	MSK	00AB	MSKTAB	CB4D
MUL.H	CB2F	MUL.L	CB49	OB1	C984
OB2	C988	OBEN	C979	OFFX	0063
PCOLOR	C92A	PL1	C85A	PL2	C8F0
PLOT	C858	PLT	C8E2	QERR	B248
R.L	C9A6	RE2	C9A5	RECHTS	C993
SCOL1	C937	SCOLOR	C931	SLINE	C86B
STORE1	CB1E	STORE2	CB1F	T1	C890
TESCOR	C879	U.O	C977	UN1	C968
UN2	C96C	UNPLOT	C855	UNTEN	C95B
USE	00FD	V	D000	VIDEO	0400
XK	0014	XKH	CB1B	XKL	CB1A
YK	CB1C	ZA	0070	ZWIS	CB20

Kommen wir zum Basiclader:

```
100 FOR I = 51200 TO 52054
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA  76, 36,200, 76, 65,200, 76, 18,201, 76, 49,201
130 DATA  76, 42,201, 76, 88,200, 76, 85,200, 76,107,200
140 DATA  76,104,200, 76, 67,202, 76, 82,202, 76,105,202
150 DATA 234,173, 17,208,141, 30,203,173, 24,208,141, 31
160 DATA 203,169, 59,141, 17,208,169, 24,141, 24,208,169
170 DATA  96,141, 36,200, 96,173, 30,203,141, 17,208,173
180 DATA  31,203,141, 24,208,169,234,141, 36,200, 76, 68
190 DATA 229,162,  0, 44,162,128,134,151, 32,253,174, 32
200 DATA 121,200, 32,148,200, 76,226,200,162,  0, 44,162
210 DATA 128, 32, 90,200, 32,253,174, 32,121,200, 76,187
220 DATA 201, 32,235,183,138,168,166, 21,192,200,176, 13
230 DATA 165, 20,224,  1,144,  6,208,  5,201, 64,176,  1
240 DATA  96, 76, 72,178,140, 28,203,141, 26,203,142, 27
250 DATA 203,133, 20,134, 21,152, 74, 74, 74,170,189, 47
260 DATA 203,133,173,138, 41,  3,170,189, 73,203,133,172
270 DATA 152, 41,  7, 24,101,172,133,172,165, 20, 41,248
280 DATA 133, 99,169, 32,  5,173,133,173, 24,165,172,101
290 DATA  99,133,172,165,173,101, 21,133,173,165, 20, 41
300 DATA   7, 73,  7,170,189, 77,203,133,171, 96,160,  0
310 DATA   8,165,171, 36,151, 48,  5, 73,255, 49,172, 44
320 DATA  17,172,145,172,165,172,133,253,165,173, 74,102
330 DATA 253, 74,102,253, 74,102,253, 41,  3,  9,  4,133
340 DATA 254,173, 29,203,145,253, 40,164,111, 96,169, 32
350 DATA 133,254,160,  0,132,253,162, 32,152,145,253,200
360 DATA 208,251,230,254,202,208,246, 76, 55,201, 32,241
370 DATA 183,142, 29,203, 96, 32,241,183,142, 29,203,162
380 DATA   3,169,  4,133,254,160,  0,132,253,132,151,173
390 DATA  29,203,145,253,200,196,151,208,249,230,254,202
400 DATA 240,  3, 16,242, 96,162,232,134,151,208,235,165
410 DATA 172, 41,  7,201,  7,240,  5, 56,169,  0,176,  4
420 DATA 169, 56,230,173,101,172,133,172,169,  0,101,173
430 DATA 133,173, 96, 48,226,165,172, 41,  7,240,  5, 24
440 DATA 169,255,144,  4,169,199,198,173,101,172,133,172
450 DATA 165,173,233,  0,133,173, 96, 70,171,144, 14,102
460 DATA 171,165,172,200, 24,105,  8,133,172,144,  2,230
470 DATA 173, 96, 16,235,  6,171,144, 14, 38,171,165,172
480 DATA 136, 56,233,  8,133,172,176,  2,198,173, 96, 72
490 DATA 173, 27,203, 74,173, 26,203,106, 74, 74,133,111
500 DATA 104, 72, 56,237, 26,203, 72,138,237, 27,203,133
510 DATA 108,176, 10,104, 73,255,105,  1, 72,169,  0,229
520 DATA 108,133,106,133,110,104,133,105,133,109,104,141
530 DATA  26,203,142, 27,203,152, 24,237, 28,203,144,  4
540 DATA  73,255,105,254,133,107,140, 28,203,102,108, 56
550 DATA 229,105,170,169,255,229,106,133,112,164,111,176
560 DATA   5, 10, 32,166,201, 56,165,109,101,107,133,109
570 DATA 165,110,233,  0,133,110,132,111, 32,226,200,232
580 DATA 208,  5,230,112,208,  1, 96,165,108,176,222, 32
```

```
590 DATA 119,201, 24,165,109,101,105,133,109,165,110,101
600 DATA 106, 80,221, 32,253,174, 32,212,225,160, 32,162
610 DATA   0,169,  0, 76,213,255, 32,253,174, 32,212,225
620 DATA 162, 63,160, 64,169,  0,133,253,169, 32,133,254
630 DATA 169,253, 76,216,255, 32,241,183,134,103, 32, 15
640 DATA 243, 32, 31,243,166,103, 32,201,255,169,255,133
650 DATA  97,169,  7,133,253,169, 28,133,151,169,  0,141
660 DATA  32,203,169, 40,141, 33,203,162,  4,189, 42,203
670 DATA  32,210,255,202, 16,247,169,  0,133, 99,133,100
680 DATA 173, 32,203,133,101,169,  0,133,254,165, 99,166
690 DATA 100,164,101, 32,148,200,160,  0,177,172,166,254
700 DATA 157, 34,203,230,101,232,134,254,228,253,208,229
710 DATA 169,  0,160,  7,166,253, 30, 34,203, 42,202, 16
720 DATA 249, 37, 97,  9,128, 32,210,255,136, 16,237,165
730 DATA  99, 24,105,  8,133, 99,144,  2,230,100,206, 33
740 DATA 203,208,181,169, 13, 32,210,255,173, 32,203, 24
750 DATA 105,  7,141, 32,203,198,151,240,  3, 76,138,202
760 DATA 169,  4,197,253,240, 12,133,253,169,  1,133,151
770 DATA 169, 15,133, 97,208,235,169, 15, 32,210,255, 76
780 DATA 204,255, 48, 48, 48, 32, 58, 32,130, 32, 88, 32
790 DATA  58, 32,143, 32, 87, 65, 80,  0, 16, 27,  8,  0
800 DATA   1,  2,  3,  5,  6,  7,  8, 10, 11, 12, 13, 15
810 DATA  16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28, 30
820 DATA  31,  0, 64,128,192,  1,  2,  4,  8, 16, 32, 64
830 DATA 128, 76, 79
840 IF S <> 104883 THEN PRINT "FEHLER IN DATAS !!" : END
850 PRINT "OK"
```

Wie ist nun dieses Graphik-Hilfsprogramm anzuwenden? Der Aufruf sämtlicher 12 Befehle erfolgt über eine SYS-Anweisung, die in einigen Fällen noch durch einige Parameter ergänzt wird. Das erscheint im ersten Moment etwas ungewöhnlich, da die Syntax des eigentlichen SYS-Befehls keine weiteren Parameter außer der Adresse zuläßt, Sie werden sich aber schnell daran gewöhnen. Am besten verfährt man, wie dies im Anwendungsbeispiel vorgeführt wird: Man definiert zunächst am Programmanfang zwölf Variablen mit den Sprungadressen zu den einzelnen Befehlen. Im Verlauf des übrigen Programms geschieht der Aufruf dann stets über diese Variablen, wobei bei Bedarf die notwendigen Parameter ergänzt werden. Eine kleine Tabelle informiert Sie über die Sprungadressen und die Syntax der verschiedenen Befehle:

SYS 51200	- Graphik einschalten
SYS 51203	- Graphik ausschalten
SYS 51206	- Graphik löschen
SYS 51209,PF*16+HF	- Farbe setzen
SYS 51212,PF*16+HF	- Plotfarbe ändern
SYS 51215,X,Y	- Punkt setzen
SYS 51218,X,Y	- Punkt löschen
SYS 51221,X1,Y1,X2,Y2	- Linie zeichnen
SYS 51224,X1,Y1,X2,Y2	- Linie löschen
SYS 51227,"name",GA	- Graphik laden
SYS 51230,"name",GA	- Graphik speichern
SYS 51233,LF	- Hardcopy S.GP-100VC

Dabei bedeuten:

PF	- Farbe eines Graphikpunktes v. 0-15
HF	- Hintergrundfarbe von 0 bis 15
X	- X-Koordinate eines Punktes (0-319)
Y	- Y-Koordinate eines Punktes (0-199)
X1/2	- X-Koordinate des Start-/Endpunktes
Y1/2	- Y-Koordinate des Start-/Endpunktes
"name"	- Filename (auch als Stringspeicher)
GA	- Geräteadresse (1 oder 8)
LF	- logische Filenummer v. OPEN LF,GA

Bevor wir direkt in unser Demonstrationsprogramm einsteigen, sollten wir kurz die wesentlichsten Dinge besprechen, wobei Sie die Details am besten durch testen und probieren herausbekommen:

Die ersten drei Befehle dürften klar sein, wobei beachtet werden sollte, daß beim Einschalten der Graphik diese nicht gelöscht werden, beim Ausschalten jedoch der Bildschirm gelöscht wird.

Beim Setzen der Farbe wird die Farbe aller Graphikpunkte und des Hintergrundes für das gesamte Graphikbild festgelegt. Wählen Sie also den Hintergrund grün (Farbcode: 5) und die Farbe der Punkte violett (Farbcode: 4), so setzen Sie PF=5 und HF=4. Bei jedem gezeichneten Punkt aber wird gleichfalls das entsprechende Farbbyte des Video-RAM gesetzt. Die jeweilige Farbe wird erstens durch den gerade besprochenen Befehl gesetzt (damit ändert sich die Farbe beim Zeichnen nicht), und zweitens durch das folgende Kommando, das lediglich besagt, daß ab jetzt alle neu gezeichneten Figuren in dieser neuen Farbe gezeichnet werden.

Ihnen stehen zwei Befehle zur Verfügung, um einen Punkt zu zeichnen und um einen Punkt zu löschen. Gleichzeitig wird im Farbram die aktuelle Farbe gesetzt.

Wollen Sie eine Linie von den Koordinaten X1,Y1 nach X2,Y2 zeichnen bzw. löschen, so verwenden Sie die beiden nächsten Befehle. Was eben zu der Farbsetzung gesagt wurde, gilt hier natürlich genauso.

Die zwei Kommandos zum Laden und Speichern von Graphik verwenden Sie bitte genauso, wie Sie Basicprogramme laden/speichern - unter Angabe des Filenamens und der Geräteadresse (GA=1 für Kassettenbetrieb, GA=8 für Diskette).

Bevor Sie eine Hardcopy auf dem Drucker Seikosha GP-100VC (oder Epson mit DATA BECKER-Interface) starten, mÜssen Sie z.B. mit OPEN 1,4 entsprechend den Druckerkanal öffnen. Anschließend folgt ein SYS HC,1 (da wir LF, die logische Filenummer als 1 gewählt haben) und ein CLOSE 1 (s. Demoprogramm).

Damit sollten die größten Unklarheiten beseitigt sein, und wir können loslegen:

```
10 REM *********************
20 REM **                 **
30 REM **  GRAPHIK-PAKET  **
40 REM **                 **
50 REM **   - D E M O -   **
60 REM **                 **
70 REM *********************
80 REM
90 REM MASCHINENROUTINEN:
100 IN=51200 : OF=51203 :REM INIT     /GRAPHIK OFF
110 GC=51206 : SC=51209 :REM GCLEAR   /SET COLOR
120 PC=51212 : PL=51215 :REM PCOLOR   /PLOT
130 UP=51218 : SL=51221 :REM UNPLOT   /SET LINE
140 CL=51224 : GL=51227 :REM CLR LINE /GLOAD
150 GS=51230 : HC=51233 :REM GSAVE    /HARDCOPY
200 REM
210 REM BEISPIELE:
220 REM **********
230 REM
300 SYS IN : REM GRAPHIK EIN
310 SYS GC : REM GRAPHIK LOESCHEN
320 SYS SC,1*16+2 : REM FARBE SETZEN
330 SYS PC,7*16+2 : REM PLOTFARBE SETZEN
340 REM
```

```
350 REM FIGUR 1:
360 REM ********
370 REM
380 FOR X=1 TO 319 STEP 4
390 SYS SL,X,50,70,X/1.6 : REM LINIEN
400 NEXT X
410 REM
420 FOR X=1 TO 5000 : NEXT X : REM WARTESCHLEIFE
430 GOSUB 2000 : SYS GC : REM GRAPHIK LOESCHEN
440 REM
450 REM FIGUR 2:
460 REM ********
470 REM
480 FOR X=1 TO 319 STEP 3
490 SYS SL,X,40,50*SIN(X/30)+100,X/1.6
500 NEXT X
510 REM
520 FOR X=1 TO 5000 : NEXT X : REM WARTESCHLEIFE
530 GOSUB 2000 : SYS GC : REM GRAPHIK LOESCHEN
540 REM
550 REM FIGUR 3:
560 REM ********
570 REM
580 FOR X=1 TO 319 STEP 2
590 SYS SL,X,40*COS(X/20)+100,50*SIN(X/30)+100,X/1.6
600 NEXT X
610 REM
620 FOR X=1 TO 5000 : NEXT X : REM WARTESCHLEIFE
630 GOSUB 2000 : SYS GC : REM GRAPHIK LOESCHEN
640 REM
1000 WAIT 198,255 : REM AUF TASTE WARTEN
1100 SYS OF : END
2000 SYS OF : REM GRAPHIK AUS
2010 PRINT "WOLLEN SIE:":PRINT
2020 PRINT "(1) - GRAPHIK LADEN"
2030 PRINT "(2) - GRAPHIK SPEICHERN"
2040 PRINT "(3) - HARDCOPY"
2050 PRINT "(4) - WEITER" : PRINT
2060 POKE 198,0 : REM TASTEN LOESCHEN
2070 WAIT 198,255 : REM AUF TASTE WARTEN
2080 GET A$
2090 ON VAL(A$) GOTO 2200,2300,2400,2500
2100 GOTO 2000
2200 REM
2210 REM GRAPHIK LADEN:
2220 REM ***************
2230 REM
2240 INPUT "FILENAME,GA";FI$,GA
2250 SYS GL,FI$,GA : REM LADEN
2260 SYS IN : REM GRAPHIK EIN
2270 SYS SC,16*3+9
```

```
2280 FOR X=1 TO 5000 : NEXT X : REM WARTEN
2290 GOTO 2000
2300 REM
2310 REM GRAPHIK SPEICHERN:
2320 REM ******************
2330 REM
2340 INPUT "FILENAME,GA";FI$,GA
2350 SYS GS,FI$,GA : REM SPEICHERN
2360 GOTO 2000
2400 REM
2410 REM HARDCOPY:
2420 REM *********
2430 REM
2440 PRINT "DRUCKER EINSCHALTEN UND TASTE DRUECKEN"
2450 POKE 198,0 : WAIT 198,255 : GET A$
2460 PRINT : PRINT "BITTE WARTEN!"
2470 OPEN 1,4 : REM DRUCKERKANAL OEFFNEN
2480 SYS HC,1 : REM HARDCOPY
2490 CLOSE 1 : REM SCHLIESSEN
2495 GOTO 2000
2500 REM
2510 REM WEITER:
2520 REM *******
2530 REM
2540 SYS IN : SYS SC,16*2+7 : RETURN
```

In diesem kleinen Demoprogramm wurden Ihnen einige Anwendungsbeispiele dargelegt. Sie können ohne weiteres Ihre eigenen Routinen mit einfügen. Viel Spaß beim Programmieren!

6. Kommentiertes Source-Listing der Supergraphik - für den Graphikfreak

Das folgende Listing ist das originale und vollständige Source-Listing der Supergraphik. Es ist durchgehend dokumentiert und gegliedert und wurde mit dem Assembler Profi-Ass von DATA BECKER erstellt.

Um Benutzern anderer Assembler das Lesen zu ermöglichen, hier einige Syntax-Besonderheiten des Profi-Ass:

.BYT	- Definieren eines Byte-Wertes
.WOR	- Definieren eines 2-Byte-Wertes
.ASC	- Definieren eines Strings
*=	- Definieren der Startadresse
=	- Wertzuweisung an Label
>label	- High-Byte des Wertes
<label	- Low-Byte des Wertes
$	- Hexadezimalzahl
%	- Dualzahl

Zu Beginn des Programmes werden einige Werte und Speicherstellen definiert, die im Laufe des Programms verwendet werden. Hier stehen Zwischenspeicher, Rechenspeicher und auch Ansprungadressen im ROM etc. Die einzelnen Adressen werden jedoch auch im Programm noch einmal an Ort und Stelle erläutert.

In jedem Fall erfahren Sie hier die Nullseitenbelegung und die Belegung der C-Seite. Für eine Übersicht über die wichtigsten Speicheradressen schauen Sie bitte in den Anhang (Kap. 7.3.3).

Nach der Initialisierung (INIT), bei der Vektoren verlegt, Farben gesetzt, die Graphik gelöscht, Parameter initialisiert und der Meldetext ausgegeben werden, finden wir die Routinen, auf die die neuen Betriebssystemvektoren zeigen:

- Zeilenübernahmeroutine
- LIST-Routine
- Befehlsauswertungsroutine
- Break/NMI
- Tastaturroutine
- Warmstart

Die ersten drei Routinen sorgen dafür, daP die neuen Supergraphikbefehle, die ebenfalls in diesem Programmteil aufgelistet sind (BEFTAB), in den BASIC-Befehlsschatz übernommen werden. Befehle werden genauso wie die originalen Befehle in Tokens übersetzt.

Wollen Sie die Startadresse eines Befehls herausfinden, müssen Sie ihn zunächst unter BEFTAB suchen und an der entsprechenden Stelle unter ADRTAB die Adresse nachschlagen.

STFLG ist die universale Routine zur Übernahme der Sekundärbefehle und des Graphikmodus für die normalen Zeichenbefehle. Hier werden die Flags FLG und FLG2 gesetzt.

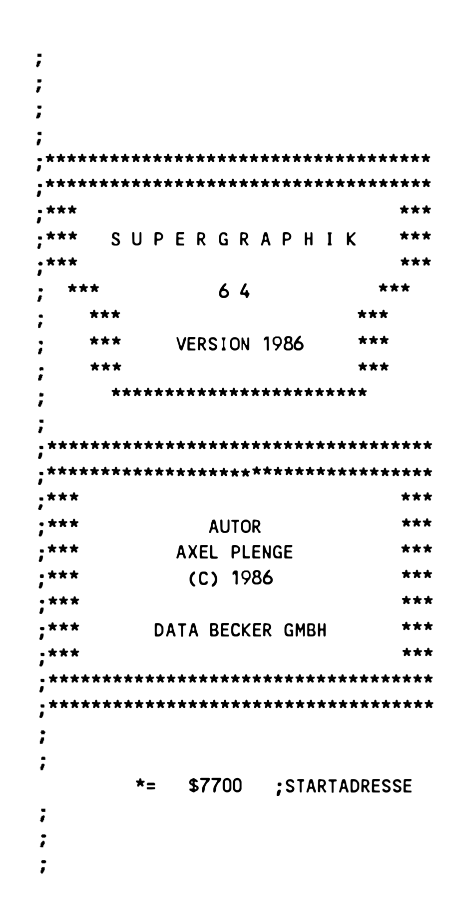
TRANS führt den Seitenwechsel beider Graphikseiten aus.

Im folgenden finden Sie nach und nach alle Supergraphikbefehle wieder. Sie sind jeweils besonders gekennzeichnet. Wenn Sie also ein Graphikphänomen verstehen wollen, so beginnen Sie am besten zunächst bei dem entsprechenden Befehlsanfang und arbeiten sich so in die tiefer liegenden Routinen ein.

Unter IRQNEU steht die neue IRQ-Routine, die verantwortlich ist für den 16-Sprite-Modus, die Text/Graphik-Mischanzeige, die Tonsteuerung und die Spritebewegung.

Besonders interessant ist die Spritebewegung, da hier dieselbe Routine verwendet wird, die zum Zeichnen einer Linie programmiert wurde. Da also das Zeichnen einer Linie und das Bewegen von Sprites zur selben Zeit stattfinden kann, die Register der Routine also für zwei verschiedene Aufgaben gleichzeitig herhalten müssen, werden sie in der IRQ-Routine zunächst zwischengespeichert, um nach der Spritebewegung wieder an die originalen Stellen zurückgebracht zu werden. Nur so stören sich die beiden Vorgänge nicht gegenseitig.

Im AnschluP an den Source-Code der Supergraphik wurden noch die Listings der Hardcopy-Routinen angehängt, die bekanntlich von $C400-$C5FF liegen und durch den Befehl HCOPY# aufgerufen werden. Darauf muP mit JSR GETBYT die Filenummer geholt werden, um an das richtige Gerät auszugeben. Sie können auf diese Weise Ihre eigene Hardcopy-Routine schreiben, an die Stelle $C400 laden und dann mit dem Befehl HCOPY# Ihre eigene Hardcopy starten.



```
                ;
                ;
                ;
                ;
                ;*************************************
                ;*************************************
                ;***                               ***
                ;***   S U P E R G R A P H I K     ***
                ;***                               ***
                ;  ***            6 4           ***
                ;    ***                      ***
                ;    ***    VERSION 1986      ***
                ;    ***                      ***
                ;      ************************
                ;
                ;*************************************
                ;*************************************
                ;***                               ***
                ;***            AUTOR              ***
                ;***         AXEL PLENGE           ***
                ;***          (C) 1986             ***
                ;***                               ***
                ;***       DATA BECKER GMBH        ***
                ;***                               ***
                ;*************************************
                ;*************************************
                ;
                ;
7700                    *=    $7700    ;STARTADRESSE
                ;
                ;
                ;
```

```
                ;ALLGEMEINE LABEL-VEREINBARUNGEN:
                ;*********************************
                ;
                ;
                ;FIXE WERTE:
                ;
                ;
00A4            TO       =   $A4     ;TOKEN
0091            AT       =   $91     ;TOKEN (ON)
00AB            MINUS    =   $AB     ;BASICCODE FUER "-"
0057            FSTBEF   =   $57     ;ERSTER BEF.
00E0            PA1G     =   $E0     ;GRAPHIKSEITE 1
00C0            PA1C     =   $C0     ;FARBSEITE 1
00D8            PA1F     =   $D8     ;FARBRAM 1
00A0            PA2G     =   $A0     ;GRAPHIKSEITE 2
00C8            PA2C     =   $C8     ;FARBSEITE 2
00CC            PA2F     =   $CC     ;FARBRAM 2
00D2            SPRDEF   =   $D2     ;SPRITEDEF $D200-$D5FF
                ;
                ;
                ;NULLSEITIGE REGISTER:
                ;
                ;
006D            OFFX     =   $6D     ;X-OFFSET
00AB            MSK      =   $AB     ;BITMASKE BEI PUNKTSETZUNG U.A.
0057            D0       =   $57     ;DIVERSE RECHENREGISTER
0058            D1       =   $58
0059            D2       =   $59
005A            D3       =   $5A
005B            D4       =   $5B
005C            D5       =   $5C
005D            D6       =   $5D
005E            D7       =   $5E
0093            E5       =   $93     ;XK/8 (HLINE)(FLAG F. LOAD/VERIFY)
005F            ZA       =   $5F     ;ZAEHLER (HLINE), U.A.
00AC            ADL      =   $AC     ;GRAPHIKADRESSE + DIVERSES
00AD            ADH      =   ADL+1
00AE            SHP      =   $AE     ;VERWENDUNG BEI DRAW + DIVERSES
0014            XK       =   $14     ;X-KOORDINATE U.A. (2 BYTE)
0002            GFLAG    =   2       ;GRAPHIKFLAG
0097            FLG      =   $97     ;NO/UN/D/PLOT-FLAG
0096            FLG2     =   $96     ;COL/UNCOLOR-FLAG
00FD            USE      =   $FD     ;DIVERSES
002D            VARSTA   =   $2D     ;VARIABLENSTARTADR.
00B8            FILENR   =   $B8     ;FILENUMMER
00B9            SECADR   =   $B9     ;SEC. ADR.
007A            PRGZG    =   $7A     ;PROGRAMMZEIGER
002B            BASSTA   =   $2B     ;PROGRAMMSTART
0090            STATUS   =   $90
0093            L.VFLG   =   $93     ;LOAD/VERIFY-FLAG
000F            HKFLG1   =   $0F     ;HOCHKOMMAFLAG 1
```

```
0008        HKFLG2   =   $08      ;HOCHKOMMAFLAG 2
0058        STKPOI   =   $58      ;STACKPOINTER FUER PAINT
            ;
            ;NULLSEITIGE REGISTER FUER FENSTER-BEFEHLE:
            ;REIHENFOLGE FESTGELEGT!
0057        WX1L     =   $57
0058        WX1H     =   $58
0059        WY1      =   $59      ;OBERE LINKE KOORDINATE
005A        WX2L     =   $5A
005B        WX2H     =   $5B
005C        WY2      =   $5C      ;UNTERE RECHTE KOORDINATE
005D        WXAL     =   $5D
005E        WXAH     =   $5E
005F        WYA      =   $5F      ;AKTUELLE KOORDINATE
0060        GRAD1L   =   $60
0061        GRAD1H   =   $61      ;AKTUELLE GRAPHIKADRESSE
0062        VRAD1L   =   $62
0063        VRAD1H   =   $63      ;AKTUELLE VIDEORAMADRESSE
0064        FRAD1L   =   $64
0065        FRAD1H   =   $65      ;AKTUELLE FARBRAMADRESSE
            ;
            ;REIHENFOLGE NICHT FESTGELEGT:
0069        GRAD2L   =   $69
006A        GRAD2H   =   $6A      ;AKT. GRAPHIKADRESSE 2. SEITE
006B        VRAD2L   =   $6B
006C        VRAD2H   =   $6C      ;AKT. VIDEORAMADRESSE 2. SEITE
006D        FRAD2L   =   $6D
006E        FRAD2H   =   $6E      ;AKT. FARBRAMADRESSE 2. SEITE
00AE        GCFLAG   =   $AE      ;GCOMB-FLAG
            ;
            ;
            ;
            ;ABSOLUTE REGISTER:
            ;
            ;
C610        ABS      =   $C610    ;BASISWERT
C610        E0       =   ABS      ;XK-LOW
C611        E1       =   ABS+1    ;XK-HIGH
C612        E2       =   ABS+2    ;YK GRAPHIKCURSOR
C613        E7       =   ABS+3    ;SCALE
C614        RT       =   ABS+4    ;ROTATIONSWINKEL
C615        E5.ZW    =   ABS+5    ;E5-ZWISCHENSP.
C616        BGRDZW   =   ABS+6    ;H-GRUNDZW-SP.
C617        GEND     =   ABS+7    ;GRAPHIK-END/START-ADRESSE
C618        GSTART   =   ABS+8    ;FUER VEKTORROUTINEN
C619        BCOL     =   ABS+9    ;H-GRUNDFARBE
C655        COLOR    =   ABS+69   ;PLOTCOLOR F.VR
C61A        COL1     =   ABS+10   ;COLOR 1
C61B        COL2     =   ABS+11   ;COLOR 2 (MC)
C61C        COL3     =   ABS+12   ;COLOR 3 (MC)
C675        COL3ZW   =   ABS+101  ;COLOR 3 ZWISCHENSP.
```

```
C61D        TCOL     =   ABS+13  ;TEXTFARBE
C61E        GFLAG2   =   ABS+14  ;SEITENFLAG
C61F        LCOLO    =   ABS+15  ;PLOTFARBE (MC)
C688        SPRIT1   =   ABS+120 ;SPRITE ENABLE
C689        SPRIT2   =   ABS+121 ;SPRITE ENABLE
C64E        NR       =   ABS+62  ;AKT. SPRITENR.
C620        VKTSAV   =   ABS+16  ;SPRUNGVEKTOREN
C62A        IRQSAV   =   VKTSAV+10  ;IRQ-VEKTOR
C62C        BRKSAV   =   IRQSAV+2   ;BRK-VEKTOR
C631        TIMEL    =   ABS+33     ;SOUNDTIMER
C634        TIMEH    =   TIMEL+3
C639        SXL      =   ABS+41  ;X-SPRITEK.LOW
C63A        SXH      =   ABS+42  ;X-SPRITEK.HIGH
C63B        SY       =   ABS+43  ;Y-SPRITEK.
C63C        Y1       =   ABS+44  ;DIVERSE KOORDINATENREGISTER:
C63E        X1L      =   ABS+46
C63F        X1H      =   ABS+47
C640        Y2       =   ABS+48
C642        X2L      =   ABS+50
C643        X2H      =   ABS+51
C644        Y3       =   ABS+52
C646        X3L      =   ABS+54
C647        X3H      =   ABS+55
C648        ZWIS     =   ABS+56  ;ZWISCHENSPEICHER FUER DIV AUFGABEN
C649        LZW      =   ABS+57  ;LGR-KOORD.-ZW.
C64B        ADRZW    =   ABS+59  ;ADL+ADH+MSK-ZWISCHENSPEICHER (3 BYTE)
C656        C1       =   ABS+70  ;DIVERSE REGISTER
C657        C2       =   ABS+71
C658        C3       =   ABS+72
C659        C4       =   ABS+73
C65A        C5       =   ABS+74
C65B        C6       =   ABS+75
C65C        C7       =   ABS+76
C65D        SAD      =   ABS+77   ;16 BYTE
C686        ZAHL     =   ABS+118  ;PUNKTEZAEHLER (2 BYTE)
C6AD        VARADR   =   ABS+157  ;VARIABLENFLAG
C676        X.SREG   =   ABS+102  ;X-SPRITEZW.(16BYTE)
C6F9        PFLAGS   =   ABS+233  ;PROZESSORZW.(16BYTES)(BIS 248)
C6AC        ZWISIQ   =   ABS+156  ;ZWISCHENSP. NUR F. IRQ!
C68C        ZLNR     =   ABS+124  ;ZEILENNR. DES BASIC-IRQ
C68E        PRGZGR   =   ABS+126  ;PROGRAMMZ. D. BASIC-IRQ
C709        SSCHRT   =   ABS+249  ;SCHRITTZAEHLER SPRITE-IRQ-16BYTES
C699        DCOLTB   =   ABS+137  ;16 BYTE FARBDRUCKER
C630        SLOW     =   ABS+32   ;ZAEHLER F. SIRQ
C637        RAST1    =   ABS+39   ;RASTERZEILEN F.
C638        RAST2    =   ABS+40   ;TEXT IN GRAPHIK
C690        PINSEL   =   ABS+128  ;PINSEL
C691        RUECK    =   ABS+129
C692        FLAGP1   =   ABS+130
C693        FLAGP2   =   ABS+131
C6C4        SKORD1   =   ABS+180  ;SPRITESATZ1 KOORD. (BIS 196)
```

```
C6D5        SKORD2  =   ABS+197  ;SPRITESATZ2 KOORD. (BIS 212)
C6E5        SCLMC1  =   ABS+213  ;MC-FARBEN (2 BYTES)-SPRITES 1
C6E7        SCLHG1  =   ABS+215  ;HGR-FARBEN (BIS 222)-SPRITES 1
C6EF        SCLMC2  =   ABS+223  ;SATZ 2
C6F1        SCLHG2  =   ABS+225  ;BIS 232
            ;REIHENFOLGE FEST:
C719        SPVEK   =   ABS+265  ;SPRITEVEKTOREN
C729        SPON1   =   ABS+281  ;ENABLE
C72A        SPON2   =   ABS+282
C72B        SPMC1   =   ABS+283  ;MC
C72C        SPMC2   =   ABS+284
C72D        SPPRI1  =   ABS+285  ;PRIORITAET
C72E        SPPRI2  =   ABS+286
C72F        SPYEX1  =   ABS+287  ;Y-EXPAND
C730        SPYEX2  =   ABS+288
C731        SPXEX1  =   ABS+289  ;X-EXPAND
C732        SPXEX2  =   ABS+290
C6AE        STKSAV  =   ABS+158  ;STACK-ZWIS (158-173)
            ;
            ;VORZUBELEGENDE ABS. REGISTER
            ;
C64F        GRMEM   =   ABS+63   ;AKT.HGR-SEITE
C650        COLMEM  =   ABS+64   ;AKT.COLOR-SEITE
C651        FABMEM  =   ABS+65   ;AKT.FARBRAM
C652        GRMM2   =   ABS+66
C653        COLMM   =   ABS+67
C654        FABMM   =   ABS+68
C66E        SRUNS1  =   ABS+94   ;SPRITELAUFMERKER
C66F        SRUNS2  =   ABS+95   ;SPRITELAUFMERKER
C670        FILON   =   ABS+96   ;FILTERREG.(S+23)
C671        WELLFO  =   ABS+97   ;WELLENFORMREG.(S+4,S)
C674        VOLUM   =   ABS+100  ;VOLUMEREG.(S+24)
D000        SREG    =   $D000    ;SPRITELAUFREG.
C68A        IRRSAV  =   ABS+122  ;COLLISIONMERKER
C68B        CFLAG   =   ABS+123  ;COLLISIONFLAG
C6AB        SPRI16  =   ABS+155
C6AA        FENSTR  =   ABS+154
C78F        FUNKTI  =   ABS+383  ;FUNKTIONSTASTENINHALT (383-511)
C694        RASFLG  =   ABS+132  ;RASTER-IRQ
C695        RSFLG1  =   ABS+133  ;OBEN/UNTEN-RASTER-FLAG
            ;
            ;FREI  -> ABS+134 - ABS+136 <-
            ;      -> ABS+174 - ABS+179 <-
            ;      -> ABS+291 - ABS+382 <-
            ;
            ;
            ;
```

```
               ;SPRUNGADRESSEN+EXTERNE REGISTER:
               ;
               ;
D000           V        =   $D000    ;VIDEOCONTROLER
D400           S        =   $D400    ;SOUNDCHIP
DD00           W        =   $DD00    ;CIA 2
0073           CHRGET   =   $73      ;NAECHSTE ZEICHEN AUS PROGRAMM
0079           CHRGOT   =   $79      ;LETZTES ZEICHEN AUS PROGRAMM
0300           VKTOLD   =   $300     ;ALTE VEKTOREN
0314           IRQ      =   $314     ;IRQ-VEKTOR
B7EB           GETCOR   =   $B7EB    ;16-BIT-ZAHL->$14/$15 + GETBYT
AEFD           CHKCOM   =   $AEFD    ;PRUEFT AUF KOMMA
B7F1           CHKGET   =   $B7F1    ;CHKCOM+GETBYT
B79E           GETBYT   =   $B79E    ;HOLT BYTEWERT->X
E1D4           GETPAR   =   $E1D4    ;FILENAMEN+GERAETENR. HOLEN
AD9E           FRMEVL   =   $AD9E    ;NAECHSTEN AUSDRUCK AUSWERTEN
AD8F           CHKSTR   =   $AD8F    ;AUF STRING TESTEN
B248           QERR     =   $B248    ;ILLIGAL QUANTITY
AF08           SERR     =   $AF08    ;SYNTAX ERROR
B658           STRERR   =   $B658    ;STRING TOO LONG
B6A3           FRESTR   =   $B6A3    ;STRINGVERWALTUNG
AD8A           FRMNUM   =   $AD8A    ;NUMERISCHEN WERT HOLEN -> FAC
B7F7           FACINT   =   $B7F7    ;FAC->INTEGER
A8A3           GOTO     =   $A8A3    ;BASIC-BEFEHL GOTO
D600           TABTAB   =   $D600    ;TABELLE UNTER ROM (RENUMBER)
D600           STADL    =   $D600    ;STACK FUER PAINT:
D700           STADH    =   $D700
D800           STMSK    =   $D800
D900           STE5     =   $D900
0302           EINGAB   =   $0302    ;EINGABE-WARTESCHLEIFEN-VEKTOR
0400           TEXRAM   =   $0400    ;TEXT-VIDEORAM
E112           BASIN    =   $E112    ;ZEICHEN-EINGABE
E10C           BASOUT   =   $E10C    ;ZEICHEN-AUSGABE
E1CC           CLOSE    =   $E1CC    ;DATEI SCHLIESSEN
E118           CHKOUT   =   $E118    ;AUSGABEKANAL OEFFNEN
FFCC           CLRCH    =   $FFCC    ;KANAELE SCHLIESSEN
E11E           CHKIN    =   $E11E    ;EINGABEKANAL OEFFNEN
B97E           OERR     =   $B97E    ;OVERFLOW-ERROR
E1C1           OPEN     =   $E1C1    ;DATEI OEFFNEN
F68F           SAVOUT   =   $F68F    ;'SAVING' AUSGEBEN
F5AF           SRCHOU   =   $F5AF    ;'SEARCHING FOR ...' AUSGEBEN
F5D2           LODOUT   =   $F5D2    ;'LOADING' AUSGEBEN
C400           HARD     =   $C400    ;POSITION DER HARDCOPY-ROUTINE
BDCD           INTOUT   =   $BDCD    ;POS. INTEGERZAHL IN A/X AUSGEBEN
FFE1           CHKSTP   =   $FFE1    ;STOP-TASTE ABFRAGEN
               ;
               ;
```

```
              ;$D-SEITENBELEGUNG (RAM):
              ;
              ;$D000-$D1FF: SPRITE-REG.
              ;$D200-$D5FF: SPRITE-DEFINITIONEN
              ;$D600-$D8FD: STACK F. RENUMBER
              ;
              ;
              ;
              ;******************************
              ;**                          **
              ;**  SUPERGRAPHIK 64  1986   **
              ;**                          **
              ;**  --- PROGRAMMSTART ---   **
              ;**                          **
              ;******************************
              ;
              ;
7700 A2 0A    INIT     LDX  #10          ;DIVERSE VEKTOREN VERBIEGEN:
7702 BD FF 02 I1.      LDA  VKTOLD-1,X ;ALTEN VEKTOR
7705 9D 1F C6          STA  VKTSAV-1,X ;MERKEN
7708 BD FB 77          LDA  VKTNEW-1,X ;NEUEN VEKTOR
770B 9D FF 02          STA  VKTOLD-1,X ;EINSETZEN
770E CA                DEX               ;(S. TABELLE)
770F D0 F1             BNE  I1.
              ;
7711 E8                INX               ;X=1
7712 BD 14 03 I0..     LDA  IRQ,X        ;IRQ/NMI-VEKTOREN
7715 9D 2A C6          STA  IRQSAV,X
7718 BD 18 03          LDA  IRQ+4,X
771B 9D 2C C6          STA  IRQSAV+2,X ;EBENFALLS VERBIEGEN
771E CA                DEX
771F 10 F1             BPL  I0..
              ;
              ;FUNKTIONSTASTEN VORBELEGEN:
              ;
7721 A0 08             LDY  #8       ;8 FUNKTIONSTASTEN
7723 84 FD             STY  USE
7725 E8                INX           ;X=0!
7726 A0 00             LDY  #0
              ;
7728 A9 10    I00      LDA  #16      ;16 BYTES PRO FUNKTIONSTASTE
772A 85 FE             STA  USE+1
772C B9 0A 78 I0...    LDA  FUNVOR,Y  ;VORBELEGUNG
772F F0 0B             BEQ  I00..     ;NAECHSTE TASTE
7731 9D 8F C7          STA  FUNKTI,X  ;KEYS VORBELEGEN
7734 E8                INX
7735 C8                INY
7736 C6 FE             DEC  USE+1
7738 D0 F2             BNE  I0...    ;BEI 16 KEINE 0!
773A F0 0B             BEQ  I000     ;UNBEDINGT
              ;
```

```
773C A9 A0    I00..    LDA  #$A0    ;REST MIT $A0 AUFFUELLEN
773E C8                INY          ;0 UEBERSPRINGEN
773F 9D 8F C7 I00...   STA  FUNKTI,X
7742 E8                INX
7743 C6 FE             DEC  USE+1
7745 D0 F8             BNE  I00...
7747 C6 FD    I000     DEC  USE
7749 D0 DD             BNE  I00
              ;
              ;
774B A9 05    I00.     LDA  #5      ;GRUEN
774D 8D 1C C6          STA  COL3
7750 8D 75 C6          STA  COL3ZW
              ;
7753 A9 00             LDA  #0
7755 A2 06             LDX  #6
7757 9D 6E C6 I0       STA  SRUNS1,X  ;SOUNDREG+SPRITELAUF
775A CA                DEX
775B 10 FA             BPL  I0
              ;
775D 78                SEI
775E 20 D2 7B          JSR  BRKGET
7761 58                CLI
              ;
7762 A9 88             LDA  #$88    ;SEITE 1
7764 8D 1E C6          STA  GFLAG2  ;NACH SEITENFLAG
7767 A9 00             LDA  #0      ;SCHWARZ
7769 8D 19 C6          STA  BCOL    ;ALS HINTERGRUND
776C 8D 16 C6          STA  BGRDZW  ;ALS HINTERGRUND SEITE 2
776F A9 01             LDA  #1
7771 8D 1F C6          STA  LCOLO   ;PUNKTFARBE = 1
7774 8D 1B C6          STA  COL2
7777 A9 00             LDA  #0
7779 8D 94 C6          STA  RASFLG  ;IRQ-FLAG LOESCHEN
777C 8D 30 C6          STA  SLOW
777F 8D 20 D0          STA  V+32    ;RAHMEN
7782 8D 21 D0          STA  V+33    ;HINTERGRUND
7785 8D 1D C6          STA  TCOL    ;TEXTFARBE
7788 8D 8A C6          STA  IRRSAV
778B 8D 8B C6          STA  CFLAG   ;COLLISIONSFLAG
778E A9 FF             LDA  #$FF
7790 8D 8A 02          STA  650     ;ALLE TASTEN REPEAT
              ;
7793 A2 05             LDX  #5
7795 BD EB 78 I0.      LDA  PAGE1G,X
7798 9D 4F C6          STA  GRMEM,X ;BEFEHLSRICHTUNGSREGISTER
779B CA                DEX
779C 10 F7             BPL  I0.
              ;
```

```
779E A9 00              LDA  #<INIT
77A0 A0 77              LDY  #>INIT   ;PROGRAMMSTARTADRESSE
77A2 85 33              STA  $33
77A4 84 34              STY  $34      ;->START STRINGS
77A6 85 35              STA  $35
77A8 84 36              STY  $36      ;->HILFZ. STRINGS
77AA 85 37              STA  $37
77AC 84 38              STY  $38      ;->BASIC-RAM ENDE
                ;
77AE A5 2D              LDA  $2D
77B0 A4 2E              LDY  $2E      ;START VARIABLEN
77B2 85 2F              STA  $2F
77B4 84 30              STY  $30      ;->START ARRAYS
77B6 85 31              STA  $31
77B8 84 32              STY  $32      ;->ENDE ARRAYS
                ;
77BA A9 00              LDA  #0
77BC 8D 15 D0           STA  V+21     ;SPRITE ENABLE
77BF 8D 88 C6           STA  SPRIT1
77C2 8D 89 C6           STA  SPRIT2
                ;
77C5 A9 0E              LDA  #14      ;HELLBLAU
77C7 8D 1A C6           STA  COL1
77CA 20 BC 8B           JSR  PCOLHG
77CD 20 4F 8B           JSR  TEXTM
                ;
77D0 A9 01              LDA  #1
77D2 85 02              STA  GFLAG    ;LGR+TEXTMODUS
77D4 20 9C 7D           JSR  TRSBEF
77D7 20 9C 7D           JSR  TRSBEF   ;FUER GEND/GSTART!
77DA 38                 SEC
77DB 20 78 83           JSR  GC2      ;GRAPHIK LOESCHEN
77DE A2 0F              LDX  #15
                ;
77E0 BD 47 78 III       LDA  DFABTB,X  ;DRUCKFARBVORWAHL
77E3 9D 99 C6           STA  DCOLTB,X  ;FUER FARBDRUCKER
77E6 CA                 DEX
77E7 10 F7              BPL  III
                ;
                ;KOPF AUSGEBEN:
                ;
77E9 A2 00              LDX  #0
77EB BD 57 78 I1        LDA  KOPF,X
77EE F0 06              BEQ  I2
77F0 20 D2 FF           JSR  $FFD2    ;BASOUT
77F3 E8                 INX
77F4 D0 F5              BNE  I1
77F6 20 44 A6 I2        JSR  $A644    ;NEW-BEFEHL
77F9 4C AE A7           JMP  $A7AE    ;INTERPRETER-SCHLEIFE
                ;
                ;
```

```
                ;TABELLE DER NEUEN VEKTORADRESSEN:
                ;**********************************
                ;
77FC 44 7C      VKTNEW   .WOR WARMST
77FE E4 7B               .WOR GETZEI
7800 F1 78               .WOR INTPRT
7802 C4 79               .WOR LIST
7804 20 7A               .WOR BEFEHL
7806 D7 93               .WOR IRQNEU
7808 AE 7B               .WOR NEWBRK
                ;
                ;
                ;
                ;FUNKTIONSTASTEN-VORBELEGUNG:
                ;*****************************
                ;
780A 4C C9      FUNVOR   .ASC "lI"
780C 0D 00               .BYT 13,0
780E 93                  .BYT 147
780F 4C C9               .ASC "lI"
7811 0D 00               .BYT 13,0
7813 44 C9               .ASC "dI"
7815 0D 00               .BYT 13,0
7817 47 CD 2C            .ASC "gM,7,161,250"
7823 0D 00               .BYT 13,0
7825 47 CD 2C            .ASC "gM,0"
7829 0D 00               .BYT 13,0
782B 47 CD 2C            .ASC "gM,8,161,250"
7837 0D 00               .BYT 13,0
7839 47 CD 2C            .ASC "gM,1"
783D 0D 00               .BYT 13,0
783F 52 C5 31            .ASC "rE10,10"
7846 00                  .BYT 0
                ;
                ;
                ;
                ;DRUCKERFARBEN-VORBELEGUNG:
                ;***************************
                ;
7847 00 07 02 DFABTB     .BYT 0,7,2,4,6,5,4,1
784F 03 02 03            .BYT 3,2,3,7,7,5,4,7
                ;
                ;
                ;
```

```
              ;STARTMELDUNG:
              ;*************
              ;
7857 92 9E 93 KOPF     .BYT 146,158,147
785A 3E 3E 3E          .ASC ">>>>>> SUPERGRAFIK CBM 64 - 1986 <<<<<<<"
7882 96 0D             .BYT 150,13
7884 20 20 20          .ASC "              DATA BECKER"
789D 0D                .BYT 13
789E 20 20 20          .ASC "       ENTWICKELT VON AXEL PLENGE"
78BF 0D                .BYT 13
78C0 1C                .BYT 28
78C1 2A 20 2A          .ASC "* * * * * * * * * * * * * * * * * * * * *"
78E8 9A 0D 00          .BYT 154,13,0
              ;
              ;
              ;
              ;HIGH-BYTES DER
              ;VIDEOSPEICHER-STARTADRESSEN:
              ;****************************
              ;
78EB E0       PAGE1G   .BYT PA1G
78EC C0       PAGE1C   .BYT PA1C
78ED D8       PAGE1F   .BYT PA1F
78EE A0       PAGE2G   .BYT PA2G
78EF C8       PAGE2C   .BYT PA2C
78F0 CC       PAGE2F   .BYT PA2F
              ;
              ;
              ;
              ;UMWANDLUNG EINER ZEILE IN INTERPRETER-CODE:
              ;******************************************
              ;
              ;QUELLZEILE IN ASCII AB: $200
              ;ZIELPUFFER AB:          $200
              ;
              ;PASS 1: EIGENE BEFEHLE-> ZWISCHENTOKEN
              ;PASS 2: BASIC 2.0-BEFEHLE TOKENISIEREN
              ;PASS 3: ZWISCHENTOKENS-> TOKENS
              ;
              ;
              ;PASS 1:
              ;EIGENE BEFEHLE ERKENNEN:
              ;EIGENE BEFEHLE WERDEN ALS TOKEN
              ;VON $01 BIS $1F UND VON $60 BIS ...
              ;UEBERSETZT. ALLE ANDEREN ZEICHEN
              ;BLEIBEN UNBEARBEITET FUER PASS 2.
              ;
              ;
```

```
78F1 A6 7A     INTPRT   LDX  PRGZG     ;QUELLPOINTER->X
78F3 A0 04              LDY  #4        ;ZIELPOINTER AUF 4
78F5 84 0F              STY  $0F       ;FLAG FUER HOCHKOMMA: "
               ;
               ;HAUPTSCHLEIFE:
               ;
78F7 BD 00 02 I3        LDA  $200,X    ;ZEICHEN AUS PUFFER (QUELLE)
78FA 30 39              BMI  I4        ;BASIC-CODE->OHNE BEARBEITUNG IN PUFFER
78FC C9 20              CMP  #" "
78FE F0 35              BEQ  I4        ;EBENSO LEERZEICHEN
               ;
7900 85 08              STA  $08       ;ZEICHEN MERKEN
7902 C9 22              CMP  #$22      ;HOCHKOMMA '"'?
7904 F0 54              BEQ  I5        ;JA->HOCHKOMMAMODUS
7906 24 0F              BIT  $0F
7908 70 2B              BVS  I4        ;HOCHKOMMAMODUS
               ;
790A C9 30              CMP  #"0"      ;<"0"?
790C 90 04              BCC  I3.A      ;JA
790E C9 3C              CMP  #$3C      ;ZIFFERN, DOPPELPUNKT...
7910 90 23              BCC  I4        ;UNBEARBEITET IN PUFFER
               ;
               ;BEFEHL IN BEFEHLSTABELLE SUCHEN:
               ;
7912 84 71     I3.A     STY  $71       ;ZIELPOINTER RETTEN
7914 A0 01              LDY  #1
7916 84 0B              STY  $0B       ;TOKENSPEICHER
7918 88                 DEY            ;Y=0
7919 86 7A              STX  PRGZG     ;X NACH QUELLPOINTERSPEICHER
791B CA                 DEX
791C C8        I3.B     INY            ;TABELLENPOINTER INC
791D E8                 INX            ;QUELLPOINTER INC
791E BD 00 02 I8        LDA  $200,X    ;ZEICHEN AUS PUFFER
7921 38                 SEC
7922 F9 92 7A           SBC  BEFTAB-1,Y  ;MIT ZEICHEN IN TAB VERGL.
7925 F0 F5              BEQ  I3.B      ;GLEICH
               ;
7927 C9 80              CMP  #$80      ;$80 ABZIEHEN FUER ENDE/KUERZEL
7929 D0 36              BNE  I6        ;NICHT GLEICH->NAECHSTEN BEFEHL TESTEN
               ;
792B A5 0B              LDA  $0B       ;TOKEN OHNE 7. BIT
792D C9 20              CMP  #$20
792F 90 02              BCC  I4.2
7931 69 3F              ADC  #$3F      ;LEGALE ZEICHEN UEBERSPRINGEN. (C=1!)
               ;
```

```
              ;TOKEN/ZEICHEN SPEICHERN:
              ;
7933 A4 71    I4.2     LDY  $71       ;ZIELPOINTER->Y
7935 E8       I4       INX            ;INC QUELLPOINTER
7936 C8                INY            ;INC ZIELPOINTER
7937 99 FB 01          STA  $1FB,Y    ;TOKEN/ZEICHEN NACH ZIELPUFFER
793A B9 FB 01          LDA  $1FB,Y    ;FLAGS SETZEN
793D F0 36             BEQ  I7        ;ZEILENENDE
              ;
793F 38                SEC
7940 E9 3A             SBC  #:"       ;DOPPELPUNKT?
7942 F0 04             BEQ  I4.A      ;JA->BEFEHLSENDE
7944 C9 49             CMP  #$49      ;BEFEHL DATA?
7946 D0 02             BNE  I4.B      ;NEIN
7948 85 0F    I4.A     STA  $0F       ;HOCHKOMMAFLAG
              ;
794A 38       I4.B     SEC
794B E9 55             SBC  #$55      ;REM?
794D D0 A8             BNE  I3        ;NEIN->NAECHSTES ZEICHEN
              ;
              ;REM/HOCHKOMMA-SCHLEIFE
794F
              ;
794F 85 08             STA  $08       ;0 NACH REM-FLAG
              ;
7951 BD 00 02 I4.C     LDA  $200,X    ;ZEICHEN HOLEN
7954 F0 DF             BEQ  I4        ;ZEILENENDE->WIEDER IN HAUPTSCHLEIFE
7956 C5 08             CMP  $08
7958 F0 DB             BEQ  I4        ;EVT. AUF DOPPELPUNKT TESTEN
795A C8       I5       INY
795B 99 FB 01          STA  $1FB,Y    ;NACH ZIELPUFFER
795E E8                INX
795F D0 F0             BNE  I4.C      ;NAECHSTES ZEICHEN (UNBEDINGT)
              ;
              ;
              ;NAECHSTEN BEFEHL IN TABELLE SUCHEN:
              ;
7961 A6 7A    I6       LDX  PRGZG     ;QUELLPOINTER
7963 E6 0B             INC  $0B       ;INC TOKENSPEICHER
7965 C8       I6.A     INY            ;INC TABELLENPOINTER
7966 B9 91 7A          LDA  BEFTAB-2,Y
7969 10 FA             BPL  I6.A      ;BIS BEFEHLSENDE SUCHEN
              ;
796B B9 92 7A          LDA  BEFTAB-1,Y  ;NAECHSTES ZEICHEN
796E D0 AE             BNE  I8    ;NOCH NICHT TABELLENENDE->WEITERSUCHEN
              ;
7970 BD 00 02          LDA  $200,X
7973 10 BE             BPL  I4.2      ;UNBEDINGT
              ;
```

```
                ;ENDE PASS 1:
                ;
7975 99 FD 01 I7        STA  $1FD,Y
                ;
                ;
                ;PASS 2:
                ;BASIC 2.0 - BEFEHLE ERKENNEN
                ;UND REGULAER TOKENISIEREN.
                ;DABEI BLEIBEN DIE SUPERGRAPHIK-
                ;ZWISCHENTOKENS UNVERAENDERT.
                ;
                ;
7978 A9 00              LDA  #0
797A 85 7A              STA  PRGZG   ;QUELLPOINTER WIEDER ZURUECK
797C 20 C1 79           JSR  INTPO   ;ZUM ALTEN INTERPRETER
                ;
                ;
                ;PASS 3:
                ;SUPERGRAPHIK-ZWISCHENTOKENS
                ;IN REGULAERE TOKENS UMWANDELN (7.BIT=1)
                ;
                ;
797F A2 00              LDX  #0      ;QUELLPOINTER
7981 A0 04              LDY  #4      ;ZIELPOINTER
7983 84 0F              STY  $0F     ;HOCHKOMMAFLAG
7985 BD 00 02 I9        LDA  $200,X  ;ZEICHEN HOLEN
7988 30 17              BMI  I10     ;BASIC 2.0 - TOKEN
798A F0 15              BEQ  I10     ;ZEILENENDE
798C C9 22              CMP  #$22    ;HOCHKOMMA
798E F0 2A              BEQ  I11     ;JA
7990 24 0F              BIT  $0F     ;HOCHKOMMAFLAG
7992 70 0D              BVS  I10     ;GESETZT
                ;
7994 C9 20              CMP  #$20    ;UNTER $20 -> ZWISCHENTOKEN
7996 90 07              BCC  I10.    ;UMWANDELN
7998 C9 60              CMP  #$60
799A 90 05              BCC  I10     ;UEBER $20, UNTER $60 -> WEITER
                ;
                ;ZWISCHENTOKEN UMWANDELN:
                ;
799C E9 40              SBC  #$40    ;C=1!
799E 18                 CLC
799F 69 D6      I10.    ADC  #FSTBEF+$7F  ;C=0!
                ;
                ;NACH ZIELPUFFER SPEICHERN:
                ;
79A1 E8         I10     INX
79A2 C8                 INY
79A3 99 FB 01           STA  $1FB,Y
79A6 B9 FB 01           LDA  $1FB,Y  ;FLAGS SETZEN
79A9 D0 DA              BNE  I9      ;KEIN ZEILENENDE->WEITER
```

```
                ;
                ;
79AB 99 FD 01          STA  $1FD,Y  ;LETZTES ZEICHEN NACH PUFFER
79AE 4C 0E A6          JMP  $A60E   ;ZEILE INS PROGRAMM EINSETZEN
                ;
                ;HOCHKOMMASCHLEIFE:
                ;
79B1 BD 00 02 I12      LDA  $200,X
79B4 F0 EB             BEQ  I10     ;ZEILENENDE
79B6 C9 22             CMP  #$22
79B8 F0 E7             BEQ  I10     ;HOCHKOMMA->WIEDER ZURUECK
79BA C8       I11      INY
79BB 99 FB 01          STA  $1FB,Y  ;ZEICHEN NACH PUFFER
79BE E8                INX
79BF D0 F0             BNE  I12     ;UNBEDINGT
                ;
                ;
                ;ALTER INTERPRETER:
                ;*******************
                ;
79C1 6C 24 C6 INTPO    JMP  (VKTSAV+4)
                ;
                ;
                ;
                ;EINEN BEFEHL/ZEICHEN AUFLISTEN:
                ;***********************************
                ;
79C4 30 03    LIST     BMI  LIS5    ;ZEICHEN/BEFEHL IN A
                ;
                ;ALTER LIST-VEKTOR:
                ;
79C6 6C 26 C6 LSTO     JMP  (VKTSAV+6)
                ;
                ;
79C9 C9 FF    LIS5     CMP  #$FF    ;PI?
79CB F0 F9             BEQ  LSTO
79CD C9 D7             CMP  #$80+FSTBEF  ;< ERSTER BEFEHL?
79CF 90 F5             BCC  LSTO    ;JA->ALTER BEFEHL
79D1 24 0F             BIT  $0F     ;HOCHKOMMA
79D3 30 F1             BMI  LSTO
                ;
                ;EIGENE BEFEHLE LISTEN:
                ;
79D5 38       LISO     SEC          ;TOKEN ZU
79D6 E9 D6             SBC  #$7F+FSTBEF  ;POINTER UMRECHNEN
79D8 AA                TAX          ;POINTER AUF TABELLE->X
79D9 84 49             STY  $49
79DB A0 FF             LDY  #$FF    ;BEFEHLSTABELLENPOINTER
                ;
```

```
              ;SUCHE BEFEHL:
              ;
79DD CA       LIS1    DEX
79DE F0 08            BEQ  LIS3    ;GEFUNDEN
              ;
              ;SUCHE BEFEHLSENDE:
              ;
79E0 C8       LIS2    INY
79E1 B9 93 7A         LDA  BEFTAB,Y
79E4 10 FA            BPL  LIS2
79E6 30 F5            BMI  LIS1    ;UNBEDINGT
              ;
              ;GEFUNDENEN BEFEHL AUSGEBEN:
              ;
79E8 C8       LIS3    INY
79E9 B9 93 7A         LDA  BEFTAB,Y
79EC 30 05            BMI  LIS4
79EE 20 47 AB         JSR  $AB47
79F1 D0 F5            BNE  LIS3
79F3 4C EF A6 LIS4    JMP  $A6EF
              ;
              ;
              ;
              ;************************
              ;**                    **
              ;**  BEFEHL: IRETURN   **
              ;**                    **
              ;************************
              ;
              ;
79F6 20 E1 92 RETIRQ  JSR  CLRCOL  ;COLLISION LOESCHEN
79F9 AD 8B C6         LDA  CFLAG
79FC 29 BF            AND  #%10111111
79FE 8D 8B C6         STA  CFLAG   ;INTERRUPTFLAG LOESCHEN
7A01 A9 FF            LDA  #$FF
7A03 85 4A            STA  $4A
7A05 20 8A A3         JSR  $A38A   ;GOSUB-SATZ AUF STACK SUCHEN
7A08 9A               TXS          ;STACK AUSRICHTEN
7A09 C9 8E            CMP  #$8E    ;IRQ-GOSUB-ERKENNUNG?
7A0B F0 03            BEQ  RET1    ;JA->OK.
              ;
7A0D 4C E0 A8         JMP  $A8E0   ;RETURN WITHOUT GOSUB
              ;
```

```
7A10 68       RET1    PLA          ;CODE FUER IRQ-GOSUB
7A11 68               PLA
7A12 85 39            STA  $39
7A14 68               PLA
7A15 85 3A            STA  $3A     ;ZEILENNUMMER
7A17 68               PLA
7A18 85 7A            STA  PRGZG
7A1A 68               PLA
7A1B 85 7B            STA  PRGZG+1 ;PROGRAMMZEIGER
7A1D 4C 6A 7A         JMP  B3      ;PROGRAMM WEITER AUSFUEHREN
              ;
              ;
              ;
              ;BEFEHLSAUSFUEHRUNG:
              ;*******************
              ;
7A20 A5 9D    BEFEHL  LDA  $9D     ;MODUS-FLAG
7A22 10 05            BPL  B0      ;PROGRAMM-MODUS
7A24 A9 00            LDA  #0      ;COLLISIONSFLAG
7A26 8D 8B C6         STA  CFLAG   ;IM DIREKTM. LOESCHEN
              ;
7A29 2C 8B C6 B0      BIT  CFLAG   ;COLLISION PRUEFEN?
7A2C 10 3C            BPL  B3      ;NEIN
7A2E 70 3A            BVS  B3      ;INTERRUPT FLAG
              ;
7A30 AD 8A C6         LDA  IRRSAV  ;COLLISION ERFOLGT?
7A33 F0 35            BEQ  B3      ;NEIN->NORMAL WEITER
              ;
              ;COLLISIONS-GOSUB AUSFUEHREN:
              ;
7A35 A9 00            LDA  #0
7A37 8D 8A C6         STA  IRRSAV  ;COLLISIONSFLAG WIEDER LOESCHEN
7A3A A9 03            LDA  #3      ;6 BYTES
7A3C 20 FB A3         JSR  $A3FB   ;PLATZ IM STACK?
              ;
7A3F A5 7B            LDA  PRGZG+1
7A41 48               PHA
7A42 A5 7A            LDA  PRGZG   ;AKTUELLEN
7A44 48               PHA          ;PROGRAMMZEIGER AUF STACK
7A45 A5 3A            LDA  $3A
7A47 48               PHA
7A48 A5 39            LDA  $39     ;AKTUELLE
7A4A 48               PHA          ;ZEILENNUMMER AUF STACK
7A4B A9 8E            LDA  #$8E    ;IRQ-GOSUB-ERKENNUNG
7A4D 48               PHA          ;AUF STACK
              ;
```

```
              ;PROGRAMM SOLL NUN HINTER DEM IF# CC THEN
              ;WEITERGEFUEHRT WERDEN:
              ;
7A4E AD 8E C6          LDA  PRGZGR
7A51 85 7A             STA  PRGZG
7A53 AD 8F C6          LDA  PRGZGR+1  ;PROGRAMMZEIGER DER
7A56 85 7B             STA  PRGZG+1   ;IF#...-DEFINITION
7A58 AD 8C C6          LDA  ZLNR
7A5B 85 39             STA  $39
7A5D AD 8D C6          LDA  ZLNR+1  ;ZEILENNUMMER DER
7A60 85 3A             STA  $3A     ;IF#...-DEFINITION SETZEN
7A62 AD 8B C6          LDA  CFLAG
7A65 09 40             ORA  #$40
7A67 8D 8B C6          STA  CFLAG   ;INTERRUPT FLAG SETZEN
              ;
              ;
              ;BEFEHL AUSFUEHREN:
              ;
7A6A 20 73 00 B3       JSR  CHRGET  ;ZEICHEN AUS PROGRAMM
7A6D C9 FF             CMP  #$FF
7A6F F0 17             BEQ  BEFO         ;PI
7A71 E9 D6             SBC  #$7F+FSTBEF  ;C=0!
7A73 90 13             BCC  BEFO    ;ALTER BEFEHL
7A75 20 7B 7A          JSR  B1      ;SUPERGRAPHIKBEFEHL AUSFUEHREN
7A78 4C AE A7          JMP  $A7AE   ;ZURUECK ZUR HAUPTSCHLEIFE
              ;
              ;
              ;SUPERGRAPAHIK-BEFEHL AUSFUEHREN:
              ;EINGANG:
              ;        A: TOKEN
              ;
7A7B 0A       B1       ASL  A         ;*2 UND 7. BIT WEG
7A7C A8                TAY            ;ALS POINTER AUF ADRESSENTABELLE->Y
7A7D B9 60 7B          LDA  ADRTAB+1,Y
7A80 48                PHA
7A81 B9 5F 7B          LDA  ADRTAB,Y ;BEFEHLSADRESSE ALS
7A84 48                PHA            ;RUECKSPRUNGADRESSE->STACK
7A85 4C 73 00          JMP  CHRGET   ;NXT ZEICHEN UND SPRUNG AUF BEFEHL
              ;
              ;
              ;BASIC 2.0 - BEFEHL AUSFUEHREN:
              ;
7A88 A5 7A    BEFO     LDA  PRGZG
7A8A D0 02             BNE  BEFO1
7A8C C6 7B             DEC  PRGZG+1
7A8E C6 7A    BEFO1    DEC  PRGZG        ;PRGZG DEC
7A90 6C 28 C6          JMP  (VKTSAV+8)  ;ALTEN BEFEHL AUSFUEHREN
              ;
              ;
              ;
```

```
                 ;ASCII-BEFEHLSTABELLE:
                 ;**********************
                 ;
                 ;BEIM LETZTEN ZEICHEN EINES BEFEHLS
                 ;IST JEWEILS ALS ENDE-MARKIERUNG
                 ;DAS 7.BIT GESETZT
                 ;
                 ;
7A93 44 49 52 BEFTAB   .ASC "directorY"
7A9C 53 50 4F          .ASC "spoweR"
7AA2 47 43 4F          .ASC "gcomB"
7AA7 44 54 41          .ASC "dtaseT"
7AAD 4D 45 52          .ASC "mergE"
7AB2 52 45 4E          .ASC "renuM"
7AB7 4B 45 D9          .ASC "keY"
7ABA 54 52 41          .ASC "tranS"
7ABF 50 4F 53          .ASC "pos"
7AC2 BD                .BYT "="+$80
7AC3 54 55 4E          .ASC "tunE"
7AC7 53 4F 55          .ASC "sounD"
7ACC 56 4F 4C          .ASC "volume"
7AD2 BD                .BYT "="+$80
7AD3 46 49 4C          .ASC "filteR"
7AD9 53 52 45          .ASC "sreaD"
7ADE 53 44 45          .ASC "sdefinE"
7AE5 53 53 45          .ASC "sseT"
7AE9 53 57 41          .ASC "swaiT"
7AEE 53 4D 4F          .ASC "smodE"
7AF3 47 4D 4F          .ASC "gmodE"
7AF8 47 43 4C          .ASC "gclеaR"
7AFE 47 4D 4F          .ASC "gmovE"
7B03 50 4C 4F          .ASC "ploT"
7B07 44 52 41          .ASC "draW"
7B0B 46 49 4C          .ASC "filL"
7B0F 46 52 41          .ASC "framE"
7B14 49 4E 56          .ASC "inverS"
7B1A 54 45 58          .ASC "texT"
7B1E 43 49 52          .ASC "circlE"
7B24 50 41 44          .ASC "paddlE"
7B2A 53 43 41          .ASC "scale"
7B2F BD                .BYT "="+$80
7B30 43 4F 4C          .ASC "color"
7B35 BD                .BYT "="+$80
7B36 53 43 4F          .ASC "scol"
7B3A BD                .BYT "="+$80
7B3B 50 43 4F          .ASC "pcol"
7B3F BD                .BYT "="+$80
7B40 47 53 41          .ASC "gsavE"
7B45 47 4C 4F          .ASC "gloaD"
7B4A 48 43 4F          .ASC "hcopY"
7B4F 49 52 45          .ASC "ireturN"
```

```
7B56 49 46              .ASC "if"
7B58 A3                 .BYT "#"+$80
7B59 50 41 49           .ASC "painT"
7B5E 00                 .BYT 0       ;TABELLENENDE
                ;
                ;
                ;
                ;TABELLE ALLER BEFEHLSADRESSEN
                ;MINUS 1 (WEGEN RTS-ANSPRUNG):
                ;******************************
                ;
7B5F 82 9D      ADRTAB  .WOR DIREKT-1
7B61 F0 90              .WOR SPRPOW-1
7B63 1D 84              .WOR GCOMB-1
7B65 1B 9B              .WOR DATAST-1
7B67 04 9B              .WOR MERGE-1
7B69 38 9B              .WOR RENUM-1
7B6B 48 9D              .WOR KEY-1
7B6D CF 82              .WOR TRANSS-1
7B6F 2F 9D              .WOR POSITI-1
7B71 01 96              .WOR TUNE-1
7B73 4D 95              .WOR SOUND-1
7B75 37 95              .WOR VOLUME-1
7B77 8F 96              .WOR FILTER-1
7B79 92 8F              .WOR SREAD-1
7B7B 0F 8F              .WOR DEFINE-1
7B7D 57 92              .WOR SSET-1
7B7F 65 93              .WOR SWAIT-1
7B81 47 90              .WOR SMODE-1
7B83 8A 8A              .WOR GMODE-1
7B85 61 83              .WOR GCLEAR-1
7B87 73 88              .WOR GMOVE-1
7B89 D8 7F              .WOR PLOT-1
7B8B 7D 8C              .WOR DRAW-1
7B8D 9A 80              .WOR FILL-1
7B8F C5 80              .WOR FRAME-1
7B91 D9 83              .WOR INVERS-1
7B93 07 97              .WOR TEXT-1
7B95 36 98              .WOR CIRC-1
7B97 19 91              .WOR PDL-1
7B99 78 8D              .WOR SCALE-1
7B9B DE 8B              .WOR BCOLOR-1
7B9D 68 8B              .WOR SCOLOR-1
7B9F 98 8B              .WOR PCOLOR-1
7BA1 27 8E              .WOR GSAVE-1
7BA3 4A 8E              .WOR GLOAD-1
7BA5 FF C3              .WOR HARD-1
7BA7 F5 79              .WOR RETIRQ-1
7BA9 C7 91              .WOR IFZ-1
7BAB DB 9D              .WOR PAINT-1
7BAD 00                 .BYT 0       ;TABELLENENDE
```

```
              ;
              ;
              ;
              ;NEUE BRK/NMI-ROUTINE:
              ;*********************
              ;
7BAE 48       NEWBRK   PHA
7BAF 8A                TXA
7BB0 48                PHA
7BB1 98                TYA
7BB2 48                PHA            ;REGISTER RETTEN
              ;
7BB3 A9 7F             LDA  #$7F
7BB5 8D 0D DD          STA  $DD0D     ;NMI-MOEGLICHKEITEN LOESCHEN
7BB8 AC 0D DD          LDY  $DD0D     ;FLAGS LESEN UND LOESCHEN
7BBB 20 BC F6          JSR  $F6BC     ;FLAG FUER STOP-TASTE SETZEN
7BBE 20 E1 FF          JSR  $FFE1     ;STOP?
7BC1 F0 03             BEQ  BRK0      ;JA
7BC3 4C 72 FE          JMP  $FE72     ;NMI FUER RS232
              ;
              ;
              ;BREAK-ROUTINE:
              ;
7BC6 20 D2 7B BRK0     JSR  BRKGET    ;INTERRUPT-VEKTOREN
7BC9 20 59 7C          JSR  GROFF     ;GRAPHIK AUS
7BCC 20 A3 FD          JSR  $FDA3     ;INTERRUPT INIT
7BCF 6C 02 A0          JMP  ($A002)   ;BASIC-WARMSTART
              ;
              ;
              ;INTERRUPT-VEKTOREN SETZEN:
              ;
7BD2 A2 01    BRKGET   LDX  #1
7BD4 BD 08 78 BRK1     LDA  VKTNEW+12,X
7BD7 9D 18 03          STA  IRQ+4,X        ;NMI-ADRESSE
7BDA BD 06 78          LDA  VKTNEW+10,X
7BDD 9D 14 03          STA  IRQ,X          ;IRQ-ADRESSE
7BE0 CA                DEX
7BE1 10 F1             BPL  BRK1
7BE3 60                RTS
              ;
              ;
              ;ZEICHEN-GET-ROUTINE:
              ;********************
              ;
7BE4 A5 C6    GETZEI   LDA  $C6       ;ZAHL DER GEDRUECKTEN TASTEN
7BE6 85 CC             STA  $CC       ;CURSOR: 0=EIN/<>0=AUS
7BE8 F0 FA             BEQ  GETZEI    ;EIN
7BEA 48                PHA
              ;
```

```
7BEB 78                SEI
7BEC A5 CF             LDA  $CF      ;CURSOR IN BLINK-PHASE?
7BEE F0 0C             BEQ  GETZA    ;NEIN!
7BF0 A5 CE             LDA  $CE      ;ZEICHEN UNTER CURSOR
7BF2 AE 87 02          LDX  $0287    ;FARBE UNTER CURSOR
7BF5 A0 00             LDY  #0
7BF7 84 CF             STY  $CF      ;CURSOR NICHT IN BLINKPHASE
7BF9 20 13 EA          JSR  $EA13    ;ZEICHEN+FARBE SETZEN
              ;
7BFC 68       GETZA    PLA           ;ANZAHL ZEICHEN
7BFD AA                TAX
7BFE CA                DEX           ;MINUS 1
7BFF BD 77 02          LDA  $0277,X  ;ZEICHEN DES TASTATURBUFF.
7C02 38                SEC
7C03 E9 85             SBC  #133     ;FUNKTIONSTASTE?
7C05 90 3A             BCC  GETZ1    ;NEIN
7C07 C9 08             CMP  #8
7C09 B0 36             BCS  GETZ1    ;NEIN
              ;
              ;FUNKTIONSTASTE:
              ;
7C0B 18                CLC
7C0C 2C E5 FD          BIT  $FDE5    ;=$04
7C0F F0 03             BEQ  GETZ6
7C11 29 03             AND  #3
7C13 38                SEC
7C14 2A       GETZ6    ROL  A        ;ASCII-WERT KORRIGIEREN
7C15 0A       GETZ2    ASL  A
7C16 0A                ASL  A
7C17 0A                ASL  A
7C18 0A                ASL  A        ;*16
7C19 A8                TAY           ;ALS POINTER
              ;
7C1A A2 00             LDX  #0
7C1C 86 C6             STX  $C6        ;ANZAHL ZEICHEN
7C1E B9 8F C7 GETZ3    LDA  FUNKTI,Y   ;BEFEHL
7C21 C9 A0             CMP  #$A0       ;LEERSTELLE?
7C23 F0 10             BEQ  GETZ7    ;JA->FERTIG
7C25 20 0C E1          JSR  BASOUT   ;AUSGEBEN
7C28 C9 0D             CMP  #$0D
7C2A F0 0F             BEQ  GETZ9    ;CR
7C2C 9D 00 02          STA  $200,X   ;IN EINGABEBUFFER
7C2F C8                INY
7C30 E8                INX
7C31 E0 10             CPX  #16
7C33 D0 E9             BNE  GETZ3    ;NAECHSTES ZEICHEN
              ;
```

```
7C35 20 62 A5 GETZ7    JSR  $A562     ;WEITER
7C38 4C 3E 7C          JMP  GETZ8
              ;
7C3B 20 CA AA GETZ9    JSR  $AACA     ;MIT 0 ABSCHLIESSEN + CR
7C3E 4C 86 A4 GETZ8    JMP  $A486     ;WEITER
              ;
7C41 6C 22 C6 GETZ1    JMP  (VKTSAV+2)  ;ALTE EINGABESCHLEIFE
              ;
              ;
              ;NEUER WARMSTART:
              ;*****************
              ;
7C44 A9 37    WARMST   LDA  #$37
7C46 85 01             STA  1         ;ROM EIN
7C48 58                CLI
7C49 8A                TXA            ;FEHLERMELDUNG MERKEN
7C4A 48                PHA
7C4B 10 04             BPL  WARM2     ;FEHLER!
7C4D 24 9D             BIT  $9D       ;DIREKTMODUS?
7C4F 30 03             BMI  WARM1     ;JA=> NICHT UMSCHALTEN
              ;
7C51 20 59 7C WARM2    JSR  GROFF     ;GRAPHIK AUS
              ;
7C54 68       WARM1    PLA
7C55 AA                TAX
7C56 6C 20 C6          JMP  (VKTSAV) ;ALTER WARMSTART
              ;
              ;
              ;GRAPHIK AUSSCHALTEN:
              ;********************
              ;
7C59 20 6C 8A GROFF    JSR  GMODEI    ;GMODE INIT
7C5C A2 00             LDX  #0
7C5E 8E 94 C6          STX  RASFLG    ;RASTER-IRQ-FLAG
7C61 8E 1A D0          STX  V+26      ;IMR
7C64 4C AB 8A          JMP  GM1       ;GMODE,0 AUSFUEHREN
              ;
              ;
              ;
              ;ZEICHENFLAGS FUER ZEICHENBEFEHLE
              ;UEBERNEHMEN:
              ;**********************************
              ;
              ;EINGANG:
              ;         A: LETZTES BEFEHLSZEICHEN
              ;
7C67 A2 00    STFLG    LDX  #0
7C69 86 97             STX  FLG       ;FLAG ZURUECKSETZEN
7C6B C9 43             CMP  #"C"      ;COLOR-SECUNDAERBEFEHL?
7C6D D0 05             BNE  STF3      ;NEIN
              ;
```

```
7C6F A2 80              LDX  #%10000000  ;7.BIT SETZEN (FLG2)
7C71 20 73 00           JSR  CHRGET  ;NAECHSTES ZEICHEN HOLEN
7C74 86 96     STF3     STX  FLG2    ;FLAG 2
               ;
7C76 C9 42              CMP  #"B"    ;BRUSH-PINSEL-SECUNDAERBEF.
7C78 D0 12              BNE  STF0    ;NEIN
               ;
7C7A A5 96              LDA  FLG2
7C7C 09 40              ORA  #%01000000  ;BIT 6 SETZEN (FLG2)
7C7E 85 96              STA  FLG2
7C80 20 73 00           JSR  CHRGET  ;NAECHSTES ZEICHEN HOLEN
7C83 20 9E B7           JSR  GETBYT  ;PINSELART -> X
7C86 8E 90 C6           STX  PINSEL  ;SPEICHERN
7C89 20 FD AE           JSR  CHKCOM  ;AUF KOMMA TESTEN
               ;
7C8C C9 54     STF0     CMP  #"T"    ;TEST-SECUNDAERBEFEHL?
7C8E D0 25              BNE  STF1    ;NEIN
               ;
7C90 A9 30              LDA  #%00110000  ;TEST-FLG SETZEN
7C92 85 97              STA  FLG
7C94 A9 00              LDA  #0
7C96 8D 86 C6           STA  ZAHL
7C99 8D 87 C6           STA  ZAHL+1  ;PUNKTEZAEHLER AUF NULL
7C9C 20 73 00           JSR  CHRGET  ;NAECHSTES ZEICHEN HOLEN
7C9F 20 8B B0           JSR  $B08B   ;VARIABLE SUCHEN
7CA2 85 49              STA  $49     ;VARIABLENADRESSE ZWISCHENSP.
7CA4 84 4A              STY  $4A
7CA6 A5 0E              LDA  $0E     ;FLAG FUER INTEGER/REAL
7CA8 8D AD C6           STA  VARADR  ;RETTEN
7CAB A5 0D              LDA  $0D     ;VARIABLENTYP
7CAD F0 03              BEQ  STF5    ;NUMERISCH->OK.
7CAF 4C AA 8F           JMP  SR1-5   ;STRING->TYP MISMATCH
               ;
7CB2 20 73 00  STF5     JSR  CHRGET  ;NAECHSTES ZEICHEN
7CB5 C9 2C     STF1     CMP  #","    ;KOMMA? => VEREINFACHUNG
7CB7 D0 04              BNE  STF1.   ;NEIN
               ;
7CB9 A9 00              LDA  #0      ;ZEICHENMODUS 0 = PLOT BEI
                                     ;FEHLENDEM PAR.
7CBB F0 14              BEQ  STF6    ;UNBEDINGT
               ;
7CBD 20 9E B7  STF1.    JSR  GETBYT  ;ZEICHENMODUS->X
7CC0 8A                 TXA          ;->A
7CC1 C9 05     STF2     CMP  #5
7CC3 90 04              BCC  STF4
7CC5 E9 04              SBC  #4      ;KORREKTUR BEI FEHLEINGABE
7CC7 B0 F8              BCS  STF2    ;UNBEDINGT
               ;
```

```
7CC9 AA       STF4     TAX
7CCA BD 55 7D          LDA  PLOTMD,X  ;HOLE PLOTMODUSFLAG
7CCD 05 97             ORA  FLG
7CCF 85 97             STA  FLG        ;MIT FLG VERKNUEPFEN
              ;
7CD1 20 FD AE STF6     JSR  CHKCOM     ;KOMMA?
              ;
              ;HOLE ZWISCHENSPEICHER NACH ADL/ADH+MSK:
              ;(GRAPHIKCURSOR)
              ;****************************************
              ;
7CD4 48       TAZ      PHA            ;AKU RETTEN
7CD5 AD 4B C6          LDA  ADRZW
7CD8 85 AC             STA  ADL
7CDA AD 4C C6          LDA  ADRZW+1
7CDD 29 1F             AND  #$1F
7CDF 0D 4F C6          ORA  GRMEM     ;ADRESSE AKTUELLE BILDSEITE
7CE2 85 AD             STA  ADH       ;NACH ADL/ADH
7CE4 AD 4D C6          LDA  ADRZW+2
7CE7 85 AB             STA  MSK
7CE9 AD 49 C6          LDA  LZW
7CEC 8D 3E C6          STA  X1L
7CEF AD 4A C6          LDA  LZW+1
7CF2 8D 3C C6          STA  Y1        ;LGR-KOORDINATEN
7CF5 68                PLA            ;AKU WIEDERHOLEN
7CF6 60                RTS
              ;
              ;ADL/ADH+MSK NACH ZWISCHENSPEICHER:
              ;(GRAPHIKCURSOR)
              ;**********************************
              ;
7CF7 A6 AC    TZA      LDX  ADL
7CF9 8E 4B C6          STX  ADRZW
7CFC A6 AD             LDX  ADH
7CFE 8E 4C C6          STX  ADRZW+1
7D01 A6 AB             LDX  MSK
7D03 8E 4D C6          STX  ADRZW+2
7D06 AE 3E C6          LDX  X1L
7D09 8E 49 C6          STX  LZW
7D0C AE 3C C6          LDX  Y1
7D0F 8E 4A C6          STX  LZW+1     ;LGR-KOORDINATEN
7D12 60                RTS
              ;
              ;
              ;USE MIT GRAPHIK-ADRESSEN LADEN:
              ;******************************
              ;
7D13 AD 50 C6 STUCOL   LDA  COLMEM
7D16 D0 08             BNE  STU       ;UNBEDINGT
              ;
```

```
7D18 AD 51 C6 STUD8     LDA  FABMEM
7D1B D0 03              BNE  STU      ;UNBEDINGT
              ;
7D1D AD 4F C6 STUGR     LDA  GRMEM
              ;
7D20 A0 00    STU       LDY  #0       ;Y=0
7D22 84 FD              STY  USE
7D24 85 FE              STA  USE+1
7D26 60                 RTS
              ;
              ;
              ;REFERENZTABELLE ZUR UMWANDLUNG
              ;EINFACHER WERTE VON 0-7 IN
              ;1-BIT-MUSTER:
              ;*******************************
              ;
7D27 01       MSKTB1    .BYT %00000001
7D28 02                 .BYT %00000010
7D29 04                 .BYT %00000100
7D2A 08                 .BYT %00001000
7D2B 10                 .BYT %00010000
7D2C 20                 .BYT %00100000
7D2D 40                 .BYT %01000000
7D2E 80                 .BYT %10000000
              ;
7D27          STAB1     =    MSKTB1
              ;
              ;
              ;REFERENZTABELLE ZUR UMWANDLUNG
              ;EINFACHER WERTE VON 0-3 IN
              ;2-BIT-MUSTER:
              ;*******************************
              ;
7D2F 03       MSKTB2    .BYT %00000011
7D30 03                 .BYT %00000011
7D31 0C                 .BYT %00001100
7D32 0C                 .BYT %00001100
7D33 30                 .BYT %00110000
7D34 30                 .BYT %00110000
7D35 C0                 .BYT %11000000
7D36 C0                 .BYT %11000000
              ;
              ;
              ;
```

```
               ;MULTIPLIKATIONSTABELLE
               ;FUER N*320:
               ;***********************
               ;
               ;HIGH-BYTES:
               ;
7D37 00 01 02 MUL.H     .BYT $00,$01,$02,$03,$05
7D3C 06 07 08           .BYT $06,$07,$08,$0A,$0B
7D41 0C 0D 0F           .BYT $0C,$0D,$0F,$10,$11
7D46 12 14 15           .BYT $12,$14,$15,$16,$17
7D4B 19 1A 1B           .BYT $19,$1A,$1B,$1C,$1E
7D50 1F                 .BYT $1F
               ;
               ;LOW-BYTES:
               ;
7D51 00 40 80 MUL.L     .BYT $00,$40,$80,$C0
               ;
7D51           STAB2    =    MUL.L
               ;
               ;
               ;
               ;
               ;ZEICHENMODUSFLAG:
               ;******************
               ;
               ;REFERENZTABELLE:
               ;
7D55 02        PLOTMD   .BYT %00000010  ;MODUS 0 (ZEICHNEN)
7D56 80                 .BYT %10000000  ;MODUS 1 (LOESCHEN)
7D57 01                 .BYT %00000001  ;MODUS 2 (INVERS)
7D58 C0                 .BYT %11000000  ;MODUS 3 (PUNKTIERT)
7D59 20                 .BYT %00100000  ;MODUS 4 (GRAPHIKCURSOR)
               ;
               ;
               ;FLG-FLAGBELEGUNG:
               ;
               ;
               ;7.BIT: 1=LOESCHEN
               ;       0=ZEICHNEN
               ;6.BIT: 1=PUNKTIERT
               ;       0=NICHT PUNKTIERT
               ;5.BIT: 1=NUR GRAPHIKCURSOR
               ;       0=NICHT NUR GC.
               ;4.BIT: 1=TEST-MODUS
               ;       0=KEIN TEST-MODUS
               ;3.BIT: %
               ;2.BIT: %
               ;1.BIT: 1=ZEICHNEN
               ;       0=NICHT ZEICHNEN
               ;0.BIT: 1=INVERS ZEICHNEN
               ;       0=NICHT INVERS ZEICHNEN
               ;
```

```
              ;
              ;
              ;
              ;
              ;FLG2-FLAGBELEGUNG:
              ;
              ;
              ;7.BIT: 1=COLOR-SECUNDAERBEFEHL
              ;6.BIT: 1=PINSEL-SECUNDAERBEFEHL
              ;
              ;
              ;
              ;GRAPHIK-SEITENWECHSEL:
              ;**********************
              ;
7D5A 78       TRANS    SEI
7D5B A5 01             LDA  1
7D5D 48                PHA
7D5E 29 FC             AND  #$FC
7D60 85 01             STA  1         ;ROM AUS
              ;
7D62 A2 20             LDX  #$20
7D64 A9 E0             LDA  #PA1G     ;GR-SEITE 1
7D66 20 20 7D          JSR  STU       ;NACH USE
7D69 A9 A0             LDA  #PA2G     ;GR-SEITE 2
7D6B 84 AC             STY  ADL
7D6D 85 AD             STA  ADH       ;NACH ADL/ADH
              ;
              ;GRAPHIKAUSTAUSCH:
              ;
7D6F B1 FD    TR1      LDA  (USE),Y ;HOLE BYTE
7D71 48                PHA          ;MERKEN
7D72 B1 AC             LDA  (ADL),Y
7D74 91 FD             STA  (USE),Y
7D76 68                PLA
7D77 91 AC             STA  (ADL),Y ;TAUSCHEN
7D79 C8                INY
7D7A D0 F3             BNE  TR1
7D7C E6 FE             INC  USE+1
7D7E E6 AD             INC  ADH
7D80 CA                DEX
7D81 D0 EC             BNE  TR1
              ;
7D83 68                PLA
7D84 85 01             STA  1         ;ROM EIN
              ;
```

```
                ;FARBSEITEN WECHSELN:
                ;
7D86 A9 C0              LDA  #PA1C   ;FARBSEITE 1
7D88 A0 C8              LDY  #PA2C   ;FARBSEITE 2
7D8A 20 D5 7D           JSR  TRCOL   ;WECHSELN
7D8D 58                 CLI
                ;
                ;EVT. MULTICOLORFARBE WECHSELN:
                ;
7D8E A5 02              LDA  GFLAG
7D90 10 22              BPL  TRSB1   ;KEIN MC
7D92 29 08              AND  #8
7D94 F0 1E              BEQ  TRSB1   ;TEXT AN
7D96 8E 21 D0           STX  V+33    ;HINTERGRUNDFARBE
7D99 20 36 8B           JSR  FRFILL  ;FARBRAM FUELLEN
                ;
                ;BEFEHLSRICHTUNGSTAUSCH:
                ;
7D9C AE 75 C6 TRSBEF    LDX  COL3ZW
7D9F AD 1C C6           LDA  COL3
7DA2 8D 75 C6           STA  COL3ZW
7DA5 8E 1C C6           STX  COL3    ;ADRESSE FARBE 3 WECHSELN
7DA8 AD 19 C6           LDA  BCOL
7DAB AE 16 C6           LDX  BGRDZW  ;HGRUNDZWISCHENSP.
7DAE 8D 16 C6           STA  BGRDZW
7DB1 8E 19 C6           STX  BCOL    ;HINTERGRUNDFARBADR. WECHSELN
                ;
7DB4 A2 02      TRSB1   LDX  #2
7DB6 BD 4F C6 TR4       LDA  GRMEM,X
7DB9 BC 52 C6           LDY  GRMM2,X
7DBC 9D 52 C6           STA  GRMM2,X
7DBF 98                 TYA
7DC0 9D 4F C6           STA  GRMEM,X
7DC3 CA                 DEX
7DC4 10 F0              BPL  TR4     ;RESTLICHE GRAPHIKADRESSEN
                ;
7DC6 AE 4F C6           LDX  GRMEM
7DC9 E8                 INX
7DCA 8E 18 C6           STX  GSTART  ;FUER HLINE-VEKTORROUTINEN
                ;
7DCD 8A                 TXA
7DCE 18                 CLC
7DCF 69 1D              ADC  #$1D
7DD1 8D 17 C6           STA  GEND    ;REG.FUER HLINE+DRAW
7DD4 60                 RTS
                ;
                ;
```

```
                ;FARBE WECHSELN:
                ;
7DD5 A2 04      TRCOL    LDX  #4
7DD7 85 FE               STA  USE+1
7DD9 84 AD               STY  ADH      ;ADRESSEN
                ;
7DDB A0 00               LDY  #0
7DDD B1 FD      TR2      LDA  (USE),Y
7DDF 48                  PHA
7DE0 B1 AC               LDA  (ADL),Y
7DE2 91 FD               STA  (USE),Y
7DE4 68                  PLA
7DE5 91 AC               STA  (ADL),Y ;WECHSELN
7DE7 C8                  INY
7DE8 D0 F3               BNE  TR2
7DEA E6 FE               INC  USE+1
7DEC E6 AD               INC  ADH
7DEE CA                  DEX
7DEF D0 EC               BNE  TR2
7DF1 60                  RTS
                ;
                ;
                ;
                ;
                ;PUNKT IN LGR ZEICHNEN:
                ;**********************
                ;
                ;UEBERGABE: X1L: X-KOORDINATE (LOW-BYTE)
                ;           X1H: X-KOORDINATE (HIGH-BYTE)
                ;           Y1 : Y-KOORDINATE
                ;
7DF2 A9 00      LPLTHP   LDA  #0
7DF2            LPLT     =    LPLTHP
7DF4 8D 48 C6            STA  ZWIS     ;FLAG FUER AUSSERHALB
7DF7 85 AB               STA  MSK
7DF9 85 AD               STA  ADH
7DFB AD 3C C6            LDA  Y1
7DFE 8D 4A C6            STA  LZW+1
7E01 85 AC               STA  ADL      ;Y NACH ADL/ADH
                ;
                ;KOORDINATEN UEBERPRUEFEN:
                ;
7E03 C9 32               CMP  #50
7E05 90 03               BCC  L7
7E07 EE 48 C6            INC  ZWIS     ;FLAG SETZEN
7E0A AD 3E C6 L7         LDA  X1L
7E0D 8D 49 C6            STA  LZW
7E10 C9 A0               CMP  #160
7E12 90 03               BCC  L8
7E14 EE 48 C6            INC  ZWIS     ;FLAG SETZEN
                ;
```

```
7E17 AD 48 C6 L8        LDA  ZWIS
7E1A D0 50              BNE  L9      ;KEIN ZEICHNEN WENN AUSSERHALB
                ;
                ;PUNKTADRESSE BERECHNEN:
                ;
7E1C 4E 3E C6           LSR  X1L     ;X/2 = SPALTENNUMMER
7E1F 26 AB              ROL  MSK     ;LOW-BIT NACH MSK
7E21 46 AC              LSR  ADL     ;Y/2 = ZEILENNUMMER
7E23 26 AB              ROL  MSK     ;LOW-BIT NACH MSK (JEDES ZEICHEN ENTHAELT 4
PUNKTE)
7E25 06 AC              ASL  ADL
7E27 06 AC              ASL  ADL
7E29 06 AC              ASL  ADL     ;ZEILENNUMMER*8
                ;
7E2B A5 AC              LDA  ADL
7E2D 48                 PHA          ;MERKEN
7E2E A5 AD              LDA  ADH
7E30 A8                 TAY
                ;
7E31 06 AC              ASL  ADL
7E33 26 AD              ROL  ADH
7E35 06 AC              ASL  ADL
7E37 26 AD              ROL  ADH     ;ZEILENNUMMER*32
                ;
7E39 18                 CLC
7E3A 68                 PLA          ;ZEILENNUMMER*8 HOLEN
7E3B 65 AC              ADC  ADL
7E3D 85 AC              STA  ADL
7E3F 98                 TYA
7E40 65 AD              ADC  ADH     ;ZEILENNUMMER*8+ZEILENNUMMER*32 =
ZEILENNUMMER*40
7E42 18                 CLC
7E43 69 04              ADC  #>$0400 ;PLUS BASISADRESSE BILDSCHIRMSPEICHER
7E45 85 AD              STA  ADH     ;GLEICH: ZEILENSTARTADRESSE
                ;
7E47 A4 AB              LDY  MSK     ;4-PUNKTE-BITMUSTER
7E49 A9 01              LDA  #1
7E4B 85 AB              STA  MSK
7E4D C0 00     L11      CPY  #0
7E4F F0 05              BEQ  L10
7E51 06 AB              ASL  MSK     ;TABELLENOFFSET HERSTELLEN:
7E53 88                 DEY          ;0001,0010,0100,1000
7E54 90 F7              BCC  L11
                ;
```

```
             ;PUNKT SETZEN:
             ;
7E56 AC 3E C6 L10     LDY  X1L      ;SPALTENNUMMER
7E59 B1 AC            LDA  (ADL),Y  ;BYTE AUS BILDSCHIRMSPEICHER
7E5B A0 00            LDY  #0
7E5D D9 F0 7E L13     CMP  L16,Y    ;BEREITS EINE PUNKTKOMBINATION?
7E60 F0 0C            BEQ  L12      ;JA
7E62 C8               INY
7E63 C0 10            CPY  #$10
7E65 90 F6            BCC  L13
7E67 A9 04            LDA  #4       ;NEIN->KEINEN PUNKT ZEICHNEN (NOPLOT)
7E69 8D 48 C6         STA  ZWIS
7E6C D0 50    L9      BNE  NLPL     ;UNBEDINGT!
             ;
7E6E A5 97    L12     LDA  FLG      ;ZEICHENFLAG TESTEN:
7E70 24 97            BIT  FLG
7E72 50 04            BVC  LPL4
             ;
7E74 49 82            EOR  #$82     ;PUNKTIERT PLOTTEN
7E76 85 97            STA  FLG
             ;
7E78 29 30    LPL4    AND  #$30
7E7A D0 4F            BNE  TLPL     ;NOPLOT/TESTPLOT
             ;
7E7C 24 97    LPL5    BIT  FLG
7E7E 10 0D            BPL  LPL2     ;PLOT
             ;
7E80 A5 AB            LDA  MSK      ;LOESCHPLOT
7E82 49 FF            EOR  #$FF
7E84 85 AB            STA  MSK
7E86 98               TYA
7E87 25 AB            AND  MSK      ;TABELLENOFFSET HERSTELLEN
7E89 A8               TAY
7E8A 18               CLC
7E8B 90 09            BCC  L12D     ;UNBEDINGT
             ;
7E8D A5 97    LPL2    LDA  FLG
7E8F 4A               LSR  A
7E90 B0 58            BCS  LIPLOT   ;INVPLOT
             ;
7E92 98               TYA           ;PLOT
7E93 05 AB            ORA  MSK      ;TABELLENOFFSET HERSTELLEN
7E95 A8               TAY
7E96 B9 F0 7E L12D    LDA  L16,Y    ;HOLE ZEICHEN AUS TAB.
7E99 24 96            BIT  FLG2     ;ZEICHENFLAG 2
7E9B 50 03            BVC  L12F
7E9D AD 90 C6         LDA  PINSEL   ;PINSEL
7EA0 AC 3E C6 L12F    LDY  X1L
7EA3 91 AC            STA  (ADL),Y  ;IN BILDSCHIRM SPEICHERN
             ;
```

```
7EA5 A5 AD              LDA   ADH
7EA7 29 03              AND   #3
7EA9 0D 51 C6           ORA   FABMEM   ;AKT. FARBRAM
7EAC 85 FE              STA   USE+1    ;FARBRAM
7EAE A5 AC              LDA   ADL
7EB0 85 FD              STA   USE
7EB2 AD 86 02           LDA   646      ;MOMENTANE TEXTFARBE
7EB5 24 96              BIT   FLG2     ;FARBFLAG
7EB7 10 03              BPL   L12E
7EB9 AD 1A C6           LDA   COL1     ;PLOTFARBE 1
7EBC 91 FD     L12E     STA   (USE),Y  ;FARBE SETZEN
               ;
7EBE AD 49 C6 NLPL      LDA   LZW      ;NOPLOT (NUR CURSOR SETZEN):
7EC1 AC 4A C6           LDY   LZW+1
7EC4 8D 3E C6           STA   X1L
7EC7 8C 3C C6           STY   Y1
7ECA 60                 RTS
               ;
7ECB 29 10     TLPL     AND   #$10
7ECD F0 EF              BEQ   NLPL     ;NOPLOT
               ;
7ECF 98                 TYA            ;TESTPLOT
7ED0 25 AB              AND   MSK
7ED2 F0 0E              BEQ   LPLTT
7ED4 AD 86 C6           LDA   ZAHL
7ED7 18                 CLC
7ED8 69 01              ADC   #1
7EDA 8D 86 C6           STA   ZAHL     ;ANZAHL DER PUNKTE ERHOEHEN
7EDD 90 03              BCC   LPLTT
7EDF EE 87 C6           INC   ZAHL+1
7EE2 A5 97     LPLTT    LDA   FLG
7EE4 29 CF              AND   #%11001111
7EE6 D0 94              BNE   LPL5
7EE8 F0 D4              BEQ   NLPL     ;UNBED.
               ;
7EEA 98        LIPLOT   TYA            ;INVERTIEREN
7EEB 45 AB              EOR   MSK
7EED A8                 TAY
7EEE B0 A6              BCS   L12D     ;UNBED.
               ;
               ;
               ;TABELLE DER ZEICHEN MIT PUNKTCHARAKTER:
               ;*****************************************
               ;
7EF0 20 7E 7B L16       .BYT $20,$7E,$7B,$61,$7C,$E2,$FF,$EC
7EF8 6C 7F 62           .BYT $6C,$7F,$62,$FC,$E1,$FB,$FE,$A0
               ;
               ;
```

```
               ;ZEICHNEN EINER LGR-LINIE:
               ;*************************
               ;
7F00 A5 02     LLINE    LDA  GFLAG    ;GRAPHIKFLAG
7F02 09 40              ORA  #$40     ;AUSSTEIGERMODE LGR
7F04 85 02              STA  GFLAG
7F06 8A                 TXA
7F07 A2 00              LDX  #0
7F09 8E 11 C6           STX  E1
7F0C 20 E4 87           JSR  HLINE    ;LINIE ZEICHNEN
7F0F A5 02              LDA  GFLAG
7F11 29 BD              AND  #%10111101
7F13 85 02              STA  GFLAG    ;AUSSTEIGERMODE WIEDER LOESCHEN
7F15 AD 10 C6           LDA  E0
7F18 AC 12 C6           LDY  E2
7F1B 8D 49 C6           STA  LZW
7F1E 8C 4A C6           STY  LZW+1
7F21 68                 PLA
7F22 8D 12 C6           STA  E2
7F25 68                 PLA
7F26 8D 11 C6           STA  E1
7F29 68                 PLA
7F2A 8D 10 C6           STA  E0       ;HGR-KOORD.WIEDERHOLEN (GRAPHIKCURSOR)
7F2D 60                 RTS
               ;
               ;
               ;
               ;LGR-VEKTORROUTINEN:
               ;*******************
               ;
               ;
               ;HINUNTER:
               ;
7F2E AC 3C C6 LUNTEN    LDY  Y1
7F31 C8                 INY
7F32 C0 32              CPY  #50
7F34 90 02              BCC  LU1
7F36 A0 00              LDY  #0
7F38 8C 3C C6 LU1       STY  Y1
7F3B 60                 RTS
               ;
               ;HOCH:
               ;
7F3C AC 3C C6 LOBEN     LDY  Y1
7F3F 88                 DEY
7F40 10 02              BPL  LO1
7F42 A0 31              LDY  #49
7F44 8C 3C C6 LO1       STY  Y1
7F47 60                 RTS
               ;
```

```
              ;RECHTS:
              ;
7F48 AC 3E C6 LRECHT   LDY  X1L
7F4B C8                INY
7F4C C0 50             CPY  #80
7F4E 90 02             BCC  LR1
7F50 A0 00             LDY  #0
7F52 8C 3E C6 LR1      STY  X1L
7F55 60                RTS
              ;
              ;LINKS:
              ;
7F56 AC 3E C6 LLINKS   LDY  X1L
7F59 88                DEY
7F5A 10 02             BPL  LLK1
7F5C A0 4F             LDY  #79
7F5E 8C 3E C6 LLK1     STY  X1L
7F61 60                RTS
              ;
              ;
              ;
              ;SPRITE-VEKTORROUTINEN:
              ;**********************
              ;
              ;
              ;HINUNTER:
              ;
7F62 A5 02    SUNTEN   LDA  GFLAG
7F64 29 02             AND  #2
7F66 F0 C6             BEQ  LUNTEN  ;LGR
7F68 EE 3B C6          INC  SY
7F6B 60                RTS          ;BEWEGE SPRITE
              ;
              ;HOCH:
              ;
7F6C A5 02    SOBEN    LDA  GFLAG
7F6E 29 02             AND  #2
7F70 F0 CA             BEQ  LOBEN   ;LGR
7F72 CE 3B C6          DEC  SY
7F75 60                RTS          ;BEWEGEN
              ;
              ;RECHTS:
              ;
7F76 A5 02    SRECHT   LDA  GFLAG
7F78 29 02             AND  #2
7F7A F0 CC             BEQ  LRECHT  ;LGR
7F7C AD 39 C6          LDA  SXL
7F7F 18                CLC
7F80 69 01             ADC  #1
7F82 8D 39 C6          STA  SXL
7F85 AD 3A C6          LDA  SXH
```

```
7F88 69 00           ADC  #0
7F8A C9 02           CMP  #2
7F8C 90 02           BCC  SR9
7F8E A9 00           LDA  #0
7F90 8D 3A C6 SR9    STA  SXH
7F93 60              RTS          ;BEW.
              ;
              ;LINKS:
              ;
7F94 A5 02    SLINKS LDA  GFLAG   ;SPRITE-LINKS
7F96 29 02           AND  #2
7F98 F0 BC           BEQ  LLINKS  ;LGR
7F9A AD 39 C6        LDA  SXL
7F9D 38              SEC
7F9E E9 01           SBC  #1
7FA0 8D 39 C6        STA  SXL
7FA3 AD 3A C6        LDA  SXH
7FA6 E9 00           SBC  #0
7FA8 B0 02           BCS  SL9
7FAA A9 01           LDA  #1
7FAC 8D 3A C6 SL9    STA  SXH
7FAF 60              RTS          ;BEW.
              ;
              ;
              ;
              ;
              ;GRAPHIKCURSOR SETZEN:
              ;*********************
              ;
              ;
7FB0 48       SETCUR PHA
7FB1 A5 97           LDA  FLG     ;ALTES FLAG RETTEN
7FB3 8D 48 C6        STA  ZWIS
7FB6 A9 20           LDA  #$20    ;NOPLOT
7FB8 85 97           STA  FLG
7FBA 68              PLA
7FBB 20 47 80        JSR  PLOT.   ;NOPLOT AUSFUEHREN
7FBE AD 48 C6        LDA  ZWIS
7FC1 85 97           STA  FLG     ;ALTES FLG WIEDERHOLEN
7FC3 60              RTS
              ;
              ;
```

```
              ;WERTZUWEISUNG AN TESTVARIABLE:
              ;*******************************
              ;
7FC4 A5 97    PLTVAR   LDA  FLG
7FC6 29 10             AND  #$10
7FC8 F0 5E             BEQ  T1       ;KEIN TEST
7FCA AD 87 C6 PLTVA.   LDA  ZAHL+1
7FCD AC 86 C6          LDY  ZAHL     ;ANZAHL KOLLISIONEN
7FD0 20 91 B3          JSR  $B391    ;NACH FIESSKOMMA
7FD3 AD AD C6          LDA  VARADR   ;INT/REAL-FLAG
7FD6 4C C2 A9          JMP  $A9C2    ;WERTZUWEISUNG
              ;
              ;
              ;
              ;*******************
              ;**              **
              ;**  PLOT-BEFEHL **
              ;**              **
              ;*******************
              ;
7FD9 20 67 7C PLOT     JSR  STFLG    ;SET FLG+FLG2(COL)
7FDC C9 A4             CMP  #TO
7FDE F0 10             BEQ  P2       ;TO-TOKEN (LINIE VOM GRAPHIKCURSOR)
7FE0 20 FF 7F          JSR  TESCOR   ;KOORDINATEN HOLEN
7FE3 20 47 80          JSR  PLOT.    ;PLOT AUSFUEHREN
              ;
7FE6 20 79 00 P1       JSR  CHRGOT
7FE9 20 F7 7C          JSR  TZA      ;KOORDINATEN ZWISCHENSPEICHERN
7FEC C9 A4             CMP  #TO
7FEE D0 D4             BNE  PLTVAR   ;TESTVARIABLENZUORDNUNG
              ;
              ;PLOT ... TO ...:
              ;
7FF0 20 73 00 P2       JSR  CHRGET
7FF3 20 FF 7F          JSR  TESCOR   ;ZWEITKOORDINATEN HOLEN
7FF6 20 5E 80          JSR  LINE.    ;LINIE ZEICHNEN
7FF9 8C 15 C6          STY  E5.ZW
7FFC 4C E6 7F          JMP  P1       ;NAECHSTES "TO" ABFRAGEN
              ;
              ;
              ;KOORDINATEN HOLEN UND UEBERPRUEFEN:
              ;***********************************
              ;
7FFF 20 EB B7 TESCOR   JSR  GETCOR
8002 8A                TXA
8003 A8                TAY
8004 A6 15             LDX  XK+1
8006 A5 02             LDA  GFLAG
8008 2C 3A E5          BIT  $E53A    ;INHALT: $04
800B D0 1B             BNE  T1       ;KEINE KONTR. BEI BIT2=1
800D 4A                LSR  A
```

```
800E B0 28                BCS  TESGR    ;LGR
8010 C0 C8                CPY  #200
8012 90 02                BCC  T2
8014 A0 C7                LDY  #199     ;FEHLEINGABE KORRIGIEREN
8016 A5 14      T2        LDA  XK
8018 24 02                BIT  GFLAG
801A 10 0D                BPL  TESHGR
                ;
                ;MC-KOORDINATEN TESTEN:
                ;
801C A6 15                LDX  XK+1     ;MC
801E D0 04                BNE  T3
8020 C9 A0                CMP  #<160
8022 90 04                BCC  T1
8024 A2 00      T3        LDX  #0
8026 A9 9F                LDA  #<159    ;KORREKTUR
8028 60         T1        RTS
                ;
                ;HGR-KOORDINATEN TESTEN:
                ;
8029 E0 01      TESHGR    CPX  #>320
802B 90 FB                BCC  T1
802D D0 04                BNE  T4       ;A: XK-LOW
802F C9 40                CMP  #<320    ;X: XK-HIGH
8031 90 F5                BCC  T1       ;Y: YK
8033 A2 01      T4        LDX  #>320    ;KORREKTUR
8035 A9 40                LDA  #<320
8037 60                   RTS
                ;
                ;LGR-KOORDINATEN TESTEN:
                ;
8038 C0 32      TESGR     CPY  #50
803A 90 02                BCC  T5
803C A0 31                LDY  #49      ;KORREKTUR
803E A5 14      T5        LDA  XK
8040 C9 50                CMP  #<80
8042 90 E4                BCC  T1
8044 A9 4F                LDA  #79      ;KORREKTUR
8046 60                   RTS
                ;
                ;
```

```
               ;EINEN PUNKT IN HGR/MC ZEICHNEN:
               ;**********************************
               ;
               ;UEBERGABE: A: X-KOORDINATE (LOW-BYTE)
               ;           X: X-KOORDINATE (HIGH-BYTE)
               ;           Y: Y-KOORDINATE
               ;
8047 48        PLOT.   PHA
8048 A5 02             LDA  GFLAG
804A 4A                LSR  A
804B B0 07             BCS  LPLOT.
804D 68                PLA
804E 20 AA 81          JSR  HPOSN   ;PUNKTADRESSE BERECHNEN
8051 4C 18 82          JMP  PLT     ;PUNKT ZEICHNEN
               ;
               ;
               ;EINEN PUNKT IN LGR ZEICHNEN:
               ;*******************************
               ;
               ;UEBERGABE: WIE OBEN
               ;
8054 68        LPLOT.  PLA          ;LOWGR-PUNKT
8055 8C 3C C6          STY  Y1
8058 8D 3E C6          STA  X1L
805B 4C F2 7D          JMP  LPLTHP  ;LGR-PLOT MIT LPOSN
               ;
               ;
               ;EINE LINIE VOM GRAPHIKCURSOR ZEICHNEN:
               ;*******************************************
               ;
               ;UEBERGABE: A: X2-KOORDINATE (LOW-BYTE)
               ;           X: X2-KOORDINATE (HIGH-BYTE)
               ;           Y: Y2-KOORDINATE
               ;
805E 48        LINE.   PHA
805F A5 97             LDA  FLG
8061 4A                LSR  A
8062 29 08             AND  #8      ;TEST?
8064 F0 0F             BEQ  L3      ;NEIN
8066 90 0D             BCC  L3      ;KEIN PLOT
               ;
8068 AD 86 C6          LDA  ZAHL    ;C=1!
806B E9 01             SBC  #1
806D 8D 86 C6          STA  ZAHL
8070 B0 03             BCS  L3
8072 CE 87 C6          DEC  ZAHL+1  ;1 DEC F. 1. PUNKT! (DOPPELT GEZEICHNET)
               ;
```

```
8075 A5 02    L3      LDA  GFLAG
8077 4A               LSR  A
8078 B0 04            BCS  LLINE.  ;LGR-LINIE
807A 68               PLA
807B 4C E4 87         JMP  HLINE
              ;
807E 68       LLINE.  PLA
807F AA               TAX
8080 AD 10 C6         LDA  E0
8083 48               PHA
8084 AD 11 C6         LDA  E1
8087 48               PHA
8088 AD 12 C6         LDA  E2
808B 48               PHA          ;KOORDINATEN HGR RETTEN
808C AD 3E C6         LDA  X1L     ;X-KL
808F 8D 10 C6         STA  E0
8092 AD 3C C6         LDA  Y1      ;Y-K
8095 8D 12 C6         STA  E2
8098 4C 00 7F         JMP  LLINE
              ;
              ;
              ;
              ;
              ;********************
              ;**                **
              ;**  BEFEHL: FILL  **
              ;**                **
              ;********************
              ;
              ;
809B 20 55 81 FILL    JSR  FINIT   ;PARAMETER+SYNTAX+NORMIEREN
              ;
809E AD 44 C6 FI0     LDA  Y3
80A1 CD 40 C6         CMP  Y2
80A4 90 1D            BCC  FI3     ;FERTIG
              ;
80A6 AD 42 C6         LDA  X2L
80A9 AE 43 C6         LDX  X2H
80AC AC 40 C6         LDY  Y2
80AF 20 B0 7F         JSR  SETCUR  ;KOORD. ALS STARTPUNKT ABSPEICHERN
80B2 AD 46 C6         LDA  X3L     ;ZIELPUNKT
80B5 AE 47 C6         LDX  X3H
80B8 AC 40 C6         LDY  Y2
80BB 20 5E 80         JSR  LINE.   ;LINIE ZEICHNEN
80BE EE 40 C6         INC  Y2      ;NAECHSTE ZEILE
80C1 D0 DB            BNE  FI0     ;UNBED.
              ;
80C3 4C C4 7F FI3     JMP  PLTVAR
              ;
              ;
              ;
```

```
              ;**********************
              ;**                  **
              ;**  BEFEHL: FRAME   **
              ;**                  **
              ;**********************
              ;
              ;
80C6 20 67 7C FRAME     JSR  STFLG
80C9 20 9E B7           JSR  GETBYT
80CC 8E 41 C6           STX  Y2+1     ;BREITE HOLEN
80CF 20 FD AE           JSR  CHKCOM
80D2 20 58 81           JSR  FINIT+3  ;S.O.
80D5 AD 41 C6           LDA  Y2+1
80D8 F0 78              BEQ  FR3      ;FERTIG
              ;
80DA AD 42 C6 FR0       LDA  X2L
80DD AE 43 C6           LDX  X2H
80E0 AC 40 C6           LDY  Y2
80E3 20 B0 7F           JSR  SETCUR   ;STARTPUNKT
80E6 AD 42 C6           LDA  X2L
80E9 AE 43 C6           LDX  X2H
80EC AC 44 C6           LDY  Y3       ;ZIELPUNKT 1
80EF 20 5E 80           JSR  LINE.    ;LINIE ZEICHNEN
80F2 AD 46 C6           LDA  X3L
80F5 AE 47 C6           LDX  X3H
80F8 AC 44 C6           LDY  Y3
80FB 20 5E 80           JSR  LINE.
80FE AD 46 C6           LDA  X3L
8101 AE 47 C6           LDX  X3H
8104 AC 40 C6           LDY  Y2
8107 20 5E 80           JSR  LINE.
810A AD 42 C6           LDA  X2L
810D AE 43 C6           LDX  X2H
8110 AC 40 C6           LDY  Y2
8113 20 5E 80           JSR  LINE.
              ;
8116 AD 46 C6           LDA  X3L      ;NAECHST INNEREN RAHMEN:
8119 38                 SEC
811A E9 01              SBC  #1
811C 8D 46 C6           STA  X3L
811F B0 03              BCS  FR9
8121 CE 47 C6           DEC  X3H      ;X3-KOORD. INNEN
              ;
8124 CE 44 C6 FR9       DEC  Y3       ;Y3-KOORD. INNEN
8127 EE 40 C6           INC  Y2       ;Y2-KOORD. INNEN
812A 18                 CLC
812B AD 42 C6           LDA  X2L
812E 69 01              ADC  #1
8130 8D 42 C6           STA  X2L
8133 90 03              BCC  FR8
8135 EE 43 C6           INC  X2H      ;X2-KOORD. INNEN
```

```
               ;
8138 CD 46 C6 FR8      CMP  X3L
813B 90 08             BCC  FR7
813D AD 43 C6          LDA  X2H
8140 CD 47 C6          CMP  X3H
8143 BO OD             BCS  FR3     ;FERTIG FALLS GESCHLOSSENES FELD
               ;
8145 AD 40 C6 FR7      LDA  Y2
8148 CD 44 C6          CMP  Y3
814B BO 05             BCS  FR3     ;FERTIG FALLS GESCHLOSSENES FELD
814D CE 41 C6          DEC  Y2+1    ;BREITE DECREMIEREN
8150 DO 88             BNE  FRO     ;NAECHST INNEREN RAHMEN ZEICHNEN
               ;
8152 4C C4 7F FR3      JMP  PLTVAR  ;TERMINIERUNG
               ;
               ;
               ;GEMEINSAME INITIALISIERUNG FILL/FRAME:
               ;
8155 20 67 7C FINIT    JSR  STFLG   ;SET FLG+FLG2(COL)
8158 20 FF 7F FINITO   JSR  TESCOR  ;KOORDINATEN HOLEN
815B 8D 42 C6          STA  X2L
815E 8E 43 C6          STX  X2H
8161 8C 40 C6          STY  Y2      ;1. KOORD.-PAAR -> X2/Y2
8164 20 79 00          JSR  CHRGOT
8167 C9 A4             CMP  #TO     ;TO-TOKEN ERWARTEN
8169 DO 3C             BNE  SERR1   ;SYNTAX ERROR
816B 20 73 00          JSR  CHRGET
816E 20 FF 7F          JSR  TESCOR  ;KOORDINATEN HOLEN
8171 8D 46 C6          STA  X3L
8174 8E 47 C6          STX  X3H
8177 8C 44 C6          STY  Y3      ;2. KOORD.-PAAR -> X3/Y3
817A CD 42 C6          CMP  X2L
               ;
               ;KLEINE KOORDINATEN ALS STARTKOORDINATEN:
               ;
817D 8A                TXA
817E ED 43 C6          SBC  X2H     ;X3-X2
8181 BO 15             BCS  FI1     ;>=0 => X3>=X2 (RICHTIG)
8183 AD 43 C6          LDA  X2H
8186 8E 43 C6          STX  X2H
8189 8D 47 C6          STA  X3H
818C AD 46 C6          LDA  X3L
818F AE 42 C6          LDX  X2L
8192 8E 46 C6          STX  X3L
8195 8D 42 C6          STA  X2L     ;AUSTAUSCH: X2 <-> X3
8198 CC 40 C6 FI1      CPY  Y2      ;Y3-Y2
819B BO 09             BCS  FI2     ;>=0'=> Y3>=Y2 (RICHTIG)
819D AD 40 C6          LDA  Y2
81A0 8C 40 C6          STY  Y2
81A3 8D 44 C6          STA  Y3      ;AUSTAUSCH: Y2 <-> Y3
81A6 60       FI2      RTS          ;Y=Y2 - X=X3L - A=Y3
```

```
               ;
81A7 4C 08 AF SERR1    JMP  SERR
               ;
               ;
               ;
               ;BERECHNUNG DER PUNKTADRESSE AUS DEN KOORDINATEN:
               ;***********************************************
               ;
               ;UEBERGABE: A: X-KOORDINATE (LOW-BYTE)
               ;           X: X-KOORDINATE (HIGH-BYTE)
               ;           Y: Y-KOORDINATE
               ;
               ;
81AA 8C 12 C6 HPOSN    STY  E2       ;Y-K
81AD 8D 10 C6          STA  E0       ;X-KL
81B0 8E 11 C6          STX  E1       ;X-KH
81B3 24 02             BIT  GFLAG    ;GRAPHIKFLAG
81B5 10 04             BPL  HPS1
               ;
81B7 0A                ASL  A        ;MC-MODE -> X-K*2
81B8 90 01             BCC  HPS1
81BA E8                INX           ;X-K.*2 (X=0 => X=1)
               ;
81BB 85 14    HPS1     STA  XK
81BD 86 15             STX  XK+1
81BF 98                TYA
81C0 4A                LSR  A
81C1 4A                LSR  A
81C2 4A                LSR  A        ;Y-K/8 = ZEILENNUMMER (FUER 8X8-PAKS)
81C3 AA                TAX
81C4 BD 37 7D          LDA  MUL.H,X ;ZEILENNUMMER*320 (HIGH-BYTE)
81C7 85 AD             STA  ADH
               ;
81C9 8A                TXA
81CA 29 03             AND  #3
81CC AA                TAX
81CD BD 51 7D          LDA  MUL.L,X ;LOW-BYTE
81D0 85 AC             STA  ADL      ;GLEICH: ZEILENSTARTADRESSE (RELATIV)
81D2 98                TYA
81D3 29 07             AND  #7
81D5 18                CLC
81D6 65 AC             ADC  ADL      ;PLUS REIHE IN 8*8-PACK
81D8 85 AC             STA  ADL      ;GLEICH: REIHENSTARTADRESSE (RELATIV)
               ;
81DA A5 14             LDA  XK
81DC 29 F8             AND  #$F8
81DE 85 6D             STA  OFFX     ;UNTERE 3 BITS DER X-K AUSBLENDEN
81E0 AD 4F C6          LDA  GRMEM    ;LAUFENDE GRAPHIKSEITE
81E3 05 AD             ORA  ADH
81E5 85 AD             STA  ADH      ;ABSOLUTE ADRESSE
               ;
```

```
81E7 18                 CLC
81E8 A5 AC              LDA  ADL
81EA 65 6D              ADC  OFFX
81EC 85 AC              STA  ADL      ;X-K HINZUADDIEREN
81EE A5 AD              LDA  ADH
81F0 65 15              ADC  XK+1
81F2 85 AD              STA  ADH      ;HIGH-BYTE
             ;
81F4 A5 14              LDA  XK       ;POSITION DES BITS INNERHALB DES BYTES
ERMITTELN:
81F6 29 07              AND  #7
81F8 49 07              EOR  #7
81FA AA                 TAX
81FB BD 27 7D           LDA  MSKTB1,X  ;ENTSPRECHENDE MASKE AUS TABELLE
81FE 24 02              BIT  GFLAG
8200 10 03              BPL  H6
8202 BD 2F 7D           LDA  MSKTB2,X  ;MC-TABELLE
8205 85 AB    H6        STA  MSK       ;ALS MASKE SPEICHERN
8207 60                 RTS
             ;
             ;
             ;
             ;EINZELNEN PUNKT BEI CURSOR ZEICHNEN:
             ;************************************
             ;
             ;
             ;RUECKSPRUNGADRESSE ELIMINIEREN:
             ;
8208 68       H7        PLA
8209 68                 PLA
820A 60                 RTS           ;ZUR UEBERGEORDNETEN ROUTINE
             ;
             ;
             ;ANSPRUNG BEI GESETZTEM AUSSTEIGERFLAG (SPRITE/LGR):
             ;
820B A5 02    SPLOT     LDA  GFLAG
820D 29 02              AND  #2
820F D0 F7              BNE  H7       ;SPRITEMODUS -> ZURUECK ZUR
INTERRUPTROUTINE
8211 08                 PHP
8212 20 F2 7D           JSR  LPLT     ;PUNKT IN LGR SETZEN
8215 4C 7D 82           JMP  NPL2
             ;
             ;
```

```
              ;REGULAERER EINSPRUNGPUNKT IN PLOT-ROUTINE:
              ;
8218 24 02    PLT      BIT  GFLAG
821A 70 EF             BVS  SPLOT    ;AUSSTEIGERFLAG GESETZT
821C 08                PHP           ;STATUSFLAGS RETTEN
              ;
821D 78                SEI
821E A5 01             LDA  1
8220 48                PHA
8221 29 FC             AND  #$FC
8223 85 01             STA  1        ;ROM AUSSCHALTEN
              ;
8225 A5 AB             LDA  MSK      ;MASKE (BITPOSITION) HOLEN
8227 24 96             BIT  FLG2     ;PINSELMODUS?
8229 50 03             BVC  PL0
822B AD 90 C6          LDA  PINSEL   ;JA->PINSEL ALS MASKE
822E 85 FE    PL0      STA  USE+1    ;MASKE
              ;
8230 A0 00             LDY  #0
8232 A5 97             LDA  FLG
8234 24 97             BIT  FLG
8236 50 04             BVC  PL4      ;6. BIT TESTEN
8238 49 82             EOR  #$82     ;PUNKTPLOT (PUNKTIERT PLOTTEN)
823A 85 97             STA  FLG
              ;
823C 29 30    PL4      AND  #%00110000  ;NOPLOT/TESTPLOT?
823E D0 64             BNE  TPL         ;JA
              ;
8240 24 97    PL5      BIT  FLG
8242 10 3D             BPL  PL2      ;PLOT+INVPLOT
              ;
8244 A5 FE    UPL      LDA  USE+1    ;LOESCHPLOT
8246 49 FF             EOR  #$FF     ;MASKE UMDREHEN
8248 31 AC             AND  (ADL),Y ;PUNKT LOESCHEN
824A 91 AC    PL1      STA  (ADL),Y ;BYTE ABSPEICHERN
824C 24 96             BIT  FLG2     ;SET COLOR?
824E 10 2A             BPL  NPL      ;KEINE FARBE SETZEN
              ;
8250 A5 AC             LDA  ADL      ;FARBSETZUNG:
8252 85 FD             STA  USE
8254 A5 AD             LDA  ADH
8256 4A                LSR  A
8257 66 FD             ROR  USE
8259 4A                LSR  A
825A 66 FD             ROR  USE
825C 4A                LSR  A
825D 66 FD             ROR  USE
825F 49 DC             EOR  #%11011100
```

```
;E000=C000*FFFF=C3FF/A000=C800*BFFF=CBFF
8261 85 FE              STA  USE+1
8263 AD 55 C6           LDA  COLOR     ;FARBE HOLEN
8266 91 FD              STA  (USE),Y   ;NACH VIDEORAM
               ;
8268 24 02              BIT  GFLAG
826A 10 0E              BPL  NPL
826C A5 FE              LDA  USE+1     ;MC
826E 29 03              AND  #3
8270 0D 51 C6           ORA  FABMEM    ;FARBRAM
8273 85 FE              STA  USE+1
8275 AD 1C C6           LDA  COL3      ;FARBE 3
8278 91 FD              STA  (USE),Y   ;SETZEN
               ;
               ;NOPLOT UND ENDE:
               ;
827A 68        NPL      PLA
827B 85 01              STA  1        ;ROM WIEDER EINSCHALTEN
827D 28        NPL2     PLP           ;CARRY+INTERR.-FLAG
827E A4 93              LDY  E5       ;Y-REG HOLEN
8280 60                 RTS           ;FERTIG
               ;
               ;
               ;PLOT UND INVPLOT:
               ;
8281 A5 97     PL2      LDA  FLG
8283 4A                 LSR  A        ;BIT 0
8284 B0 3F              BCS  IPL      ;INVPLOT
8286 A5 FE              LDA  USE+1
8288 24 02              BIT  GFLAG
828A 30 04              BMI  PL3      ;MC
828C 11 AC              ORA  (ADL),Y
828E 90 BA              BCC  PL1      ;UNBEDINGT
               ;
               ;MC-PLOT:
               ;
8290 AC 1F C6  PL3      LDY  LCOLO     ;PLOTFARBE (1-3)
8293 48                 PHA            ;PUNKTMASKE RETTEN
8294 39 CC 82           AND  COLTB1,Y  ;FARBMASKE HOLEN
8297 85 FD              STA  USE       ;RETTEN
8299 68                 PLA            ;PUNKTMASKE
829A 49 FF              EOR  #$FF
829C A0 00              LDY  #0
829E 31 AC              AND  (ADL),Y   ;BETREFFENDE BITS LOESCHEN
82A0 05 FD              ORA  USE       ;UND ENTSPRECHEND FARBMASKE VERAENDERN
82A2 90 A6              BCC  PL1       ;UNBED.
               ;
```

```
              ;TESTPLOT UND NOPLOT:
              ;(PLOT MIT PUNKTTEST UND NUR CURSOR SETZEN)
              ;
82A4 29 10    TPL      AND  #$10
82A6 F0 D2             BEQ  NPL      ;NOPLOT
82A8 A5 FE             LDA  USE+1    ;TESTPLOT
82AA 31 AC             AND  (ADL),Y ;PUNKT GESETZT?
82AC F0 0E             BEQ  PLTT     ;NEIN
              ;
82AE AD 86 C6          LDA  ZAHL     ;JA...
82B1 18                CLC
82B2 69 01             ADC  #1
82B4 8D 86 C6          STA  ZAHL
82B7 90 03             BCC  PLTT
82B9 EE 87 C6          INC  ZAHL+1   ;INC PUNKTEZAEHLER
              ;
82BC A5 97    PLTT     LDA  FLG
82BE 29 CF             AND  #%11001111
82C0 F0 B8             BEQ  NPL      ;NOPLOT
82C2 4C 40 82          JMP  PL5      ;PLOT/INVPLOT ETC.
              ;
              ;INVERTIERT PLOTTEN (INVPLOT):
              ;
82C5 A5 FE    IPL      LDA  USE+1    ;IPL
82C7 51 AC             EOR  (ADL),Y ;INVERTIEREN
82C9 4C 4A 82          JMP  PL1
              ;
              ;
              ;FARBBITMUSTERTABELLE FUER MC:
              ;
82CC 00 55 AA COLTB1   .BYT %00000000,%01010101,%10101010,%11111111
              ;
              ;
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: TRANS  **
              ;**                 **
              ;*********************
              ;
              ;
82D0 20 9E B7 TRANSS   JSR  GETBYT   ;0=GROSS/GRAPHIK//1=GROSS/KLEIN
              ;
82D3 78                SEI
82D4 A5 01             LDA  1
82D6 48                PHA
82D7 A9 33             LDA  #$33
82D9 85 01             STA  1          ;ROM AUS
              ;
```

```
82DB 8A                 TXA             ;MODUS->A
82DC 4A                 LSR  A          ;NACH CARRY
82DD 08                 PHP             ;ALS FLAG
               ;
82DE A9 00              LDA  #<TEXRAM   ;STARTADRESSE-TEXT
82E0 A0 04              LDY  #>TEXRAM
82E2 85 FD              STA  USE
82E4 84 FE              STY  USE+1      ;NACH USE
               ;
82E6 AD 4F C6           LDA  GRMEM      ;STARTADRESSE GRAPHIK
82E9 A0 00              LDY  #0
82EB 85 AD              STA  ADH
82ED 84 AC              STY  ADL        ;NACH ADL/ADH
               ;
               ;UEBERTRAGUNG:
               ;
82EF 84 58     TRA0     STY  D0+1      ;MATRIXADRESSE (HIGH-BYTE)
82F1 B1 FD              LDA  (USE),Y   ;LADE ZEICHEN
               ;
               ;MATRIXADRESSE ERRECHNEN:
               ;
82F3 A2 03              LDX  #3
82F5 0A        TRA1     ASL  A
82F6 26 58              ROL  D0+1     ;*8
82F8 CA                 DEX
82F9 D0 FA              BNE  TRA1
82FB 85 57              STA  D0       ;RELATIVE MATRIXADRESSE (LOW-BYTE)
               ;
82FD A5 58              LDA  D0+1
82FF 28                 PLP           ;ZEICHENSATZFLAG
8300 08                 PHP
8301 B0 03              BCS  TRA2
8303 09 D0              ORA  #$D0     ;BASISADRESSE - HIGH
8305 2C                 .BYT $2C      ;NAECHSTEN BEFEHL UEBERSPRINGEN
8306 09 D8     TRA2     ORA  #$D8
8308 85 58              STA  D0+1     ;=ZEICHENADRESSE-HIGH
               ;
               ;ZEICHENMATRIX NACH GRAPHIK:
               ;
830A A0 07              LDY  #7
830C B1 57     TRA3     LDA  (D0),Y   ;MATRIXBYTES
830E 91 AC              STA  (ADL),Y ;IN GRAPHIK UEBERTRAGEN
8310 88                 DEY
8311 10 F9              BPL  TRA3
               ;
```

```
                  ;POINTER INCREMIEREN
8313
                  ;
8313 C8                  INY            ;Y=0
8314 18                  CLC
8315 A5 AC               LDA  ADL
8317 69 08               ADC  #8
8319 85 AC               STA  ADL       ;GRAPHIKPOINTER + 8
831B 90 02               BCC  TRA4
831D E6 AD               INC  ADH
                  ;
831F E6 FD        TRA4   INC  USE       ;TEXTPOINTER + 1
8321 D0 CC               BNE  TRA0      ;WEITER
8323 E6 FE               INC  USE+1
8325 A5 FE               LDA  USE+1
8327 C9 08               CMP  #>TEXRAM+1024
8329 D0 C4               BNE  TRA0      ;NICHT FERTIG!
                  ;
832B 28                  PLP
832C 68                  PLA
832D 85 01               STA  1         ;ROM AN
832F 58                  CLI
                  ;
                  ;TEXTFARBE UEBERTRAGEN:
                  ;
8330 20 13 7D            JSR  STUCOL    ;VIDEORAMADR.->USE
8333 84 97               STY  FLG       ;Y=0
8335 84 AC               STY  ADL
8337 A9 D8               LDA  #$D8      ;FARBRAM
8339 85 AD               STA  ADH
                  ;
833B A2 03               LDX  #3
833D B1 AC        TRA5   LDA  (ADL),Y   ;FARBRAM
833F 0A                  ASL  A
8340 0A                  ASL  A
8341 0A                  ASL  A
8342 0A                  ASL  A
8343 85 57               STA  D0
8345 B1 FD               LDA  (USE),Y   ;IN VIDEORAM UEBERTRAGEN
8347 29 0F               AND  #$0F
8349 05 57               ORA  D0
834B 91 FD               STA  (USE),Y
834D C8                  INY
834E C4 97               CPY  FLG
8350 D0 EB               BNE  TRA5
8352 E6 AD               INC  ADH
8354 E6 FE               INC  USE+1
8356 CA                  DEX
8357 F0 03               BEQ  TRA6
8359 10 E2               BPL  TRA5
835B 60                  RTS
```

```
               ;
               ;
835C A2 E8     TRA6     LDX  #$E8     ;SPRITEVEKTOREN SICHERN
835E 86 97              STX  FLG
8360 D0 DB              BNE  TRA5     ;UNBED.
               ;
               ;
               ;
               ;
               ;
               ;**********************
               ;**                  **
               ;**  BEFEHL: GCLEAR  **
               ;**                  **
               ;**********************
               ;
               ;
8362 A9 00     GCLEAR   LDA  #0
8364 85 97              STA  FLG      ;LOESCH-FLAG VORBELEGEN
8366 20 79 00           JSR  CHRGOT   ;LETZTES ZEICHEN HOLEN
8369 38                 SEC           ;FLAG FUER ALTE ROUTINE
836A F0 0C              BEQ  GC2      ;NUR GCLEAR
836C C9 2C              CMP  #","     ;KOMMA?
836E F0 05              BEQ  GC1      ;JA->LOESCH-FLAG AUSGELASSEN
               ;
8370 20 9E B7           JSR  GETBYT   ;HOLE LOESCHFLAG->X
8373 86 97              STX  FLG      ;NACH FLG
               ;
8375 20 12 85  GC1      JSR  WINIT    ;FENSTER-PARAMETER HOLEN, INITIALISIERUNG
               ;
8378 78        GC2      SEI
8379 A5 01              LDA  1
837B 48                 PHA
837C 29 FC              AND  #$FC
837E 85 01              STA  1        ;ROM AUS
8380 B0 31              BCS  GCLV     ;VOLLEN BILDSCHIRM LOESCHEN
               ;
               ;BILDSCHIRMFENSTER LOESCHEN:
               ;***************************
               ;
8382 A0 00              LDY  #0
8384 20 57 86  GC3      JSR  GRIGHT   ;HOLE ADRESSE+MASKE DES NAECHSTEN BYTE
8387 90 1C              BCC  GC4      ;FERTIG
8389 48                 PHA
838A 25 97              AND  FLG      ;GCLEAR-FLAG KAPPEN
838C 8D 48 C6           STA  ZWIS     ;UND MERKEN
838F 68                 PLA           ;MASKE
8390 49 FF              EOR  #$FF     ;MASKE UMDREHEN (ZU LOESCHENDE BITS=0)
8392 31 60              AND  (GRAD1L),Y  ;BITS IN SPEICHER AUSBLENDEN
8394 0D 48 C6           ORA  ZWIS        ;GCLEAR-FLAG
8397 91 60              STA  (GRAD1L),Y  ;UND ZURUECK
```

```
8399 AD 55 C6          LDA  COLOR       ;FARBE
839C 91 62             STA  (VRAD1L),Y  ;VIDEORAM LOESCHEN
839E AD 1C C6          LDA  COL3        ;FARBE 3
83A1 91 64             STA  (FRAD1L),Y  ;FARBRAM LOESCHEN
83A3 B0 DF             BCS  GC3         ;UNBEDINGT->NAECHSTES BYTE
                 ;
83A5 68          GC4   PLA
83A6 85 01             STA  1           ;ROM EIN
83A8 58                CLI
                 ;
83A9 18                CLC
83AA AD 4F C6          LDA  GRMEM   ;AKT. GRAPHIKSEITE
83AD 69 1E             ADC  #$1E
83AF 8D 17 C6          STA  GEND    ;GEND WIEDER HERSTELLEN
83B2 60                RTS
                 ;
                 ;
                 ;VOLLES BILDSCHIRMFENSTER LOESCHEN:
                 ;***********************************
                 ;
83B3 20 1D 7D GCLV     JSR  STUGR   ;GRAPHIKADRESSE -> USE
83B6 A2 20             LDX  #$20    ;ZAEHLER
83B8 A5 97             LDA  FLG     ;LOESCH-FLAG
83BA 91 FD       GC5   STA  (USE),Y ;SPEICHER LOESCHEN
83BC C8                INY
83BD D0 FB             BNE  GC5
83BF E6 FE             INC  USE+1   ;HIGH-BYTE ERHOEHEN
83C1 CA                DEX
83C2 D0 F6             BNE  GC5
                 ;
83C4 68                PLA
83C5 85 01             STA  1       ;ROM WIEDER AN
83C7 58                CLI
                 ;
83C8 86 AB             STX  MSK     ;ALLE BITS V. VIDEORAM
83CA 20 BC 8B          JSR  PCOLHG  ;FARBE FUER HGR SETZEN
83CD 24 02             BIT  GFLAG   ;GRAPHIKFLAG
83CF 10 06             BPL  GC6     ;HGR
83D1 20 CA 8B          JSR  PCOLMC  ;FARBE FUER MC SETZEN
83D4 20 36 8B          JSR  FRFILL  ;FARBE 3
83D7 4C 0D 8C GC6      JMP  SCOL1   ;VIDEORAM LOESCHEN
                 ;
                 ;
                 ;
```

```
              ;**********************
              ;**                  **
              ;**  BEFEHL: INVERS  **
              ;**                  **
              ;**********************
              ;
              ;
83DA A9 FF    INVERS   LDA  #$FF
83DC 85 97             STA  FLG      ;INVERS-FLAG VORBELEGEN
83DE 20 79 00          JSR  CHRGOT   ;LETZTES ZEICHEN
83E1 38                SEC           ;FLAG FUER VOLLES FENSTER
83E2 F0 0C             BEQ  INV2     ;NUR INVERS
83E4 C9 2C             CMP  #","     ;KOMMA?
83E6 F0 05             BEQ  INV1     ;JA->FLAG AUSGELASSEN
              ;
83E8 20 9E B7          JSR  GETBYT   ;FLAG->X
83EB 86 97             STX  FLG      ;NACH FLG
              ;
83ED 20 12 85 INV1     JSR  WINIT    ;FENSTER-PARAMETER HOLEN+INITIALISIERUNG
              ;
83F0 78       INV2     SEI
83F1 A5 01             LDA  1
83F3 48                PHA
83F4 29 FC             AND  #$FC
83F6 85 01             STA  1
83F8 B0 0F             BCS  INVV     ;INVERTIERE VOLLES FENSTER
              ;
              ;BILDSCHIRMFENSTER INVERTIEREN:
              ;*******************************
              ;
83FA A0 00             LDY  #0
83FC 20 57 86 INV3     JSR  GRIGHT       ;NAECHSTES BYTE+MASKE
83FF 90 A4             BCC  GC4          ;FERTIG
8401 25 97             AND  FLG          ;MASKE MIT INV-MASKE KOMBINIEREN
8403 51 60             EOR  (GRAD1L),Y ;INVERTIEREN
8405 91 60             STA  (GRAD1L),Y ;UND ZURUECK
8407 B0 F3             BCS  INV3         ;UNBEDINGT
              ;
              ;
              ;VOLLES BILDSCHIRMFENSTER INVERTIEREN:
              ;*************************************
              ;
8409 20 1D 7D INVV     JSR  STUGR    ;GRAPHIKADRESSE NACH USE
840C A2 20             LDX  #$20
840E B1 FD    INV4     LDA  (USE),Y
8410 45 97             EOR  FLG      ;INVERTIEREN
8412 91 FD             STA  (USE),Y
8414 C8                INY
8415 D0 F7             BNE  INV4
8417 E6 FE             INC  USE+1
8419 CA                DEX
```

```
841A D0 F2          BNE  INV4
              ;
841C F0 87          BEQ  GC4     ;UNBEDINGT->ENDE
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: GCOMB  **
              ;**                 **
              ;*********************
              ;
              ;
841E 20 9E B7 GCOMB JSR  GETBYT  ;MODUS NACH X
8421 8A             TXA          ;NACH A
8422 29 04          AND  #%00000100
8424 85 97          STA  FLG     ;COLOR-FLAG
8426 8A             TXA
8427 29 03          AND  #3
8429 AA             TAX            ;POINTER AUF TABELLE
842A BD 05 85       LDA  COMBTB,X  ;LADE BEFEHLSCODE
842D 8D F5 84       STA  GC06      ;INS PROGRAMM SCHREIBEN
8430 8A             TXA
8431 0A             ASL  A
8432 AA             TAX            ;MODUS * 2
8433 BD 09 85       LDA  COMBT2,X  ;SPRUNGADRESSE
8436 8D 94 84       STA  GCOJ+1    ;INS PROGRAMM
8439 BD 0A 85       LDA  COMBT2+1,X
843C 8D 95 84       STA  GCOJ+2
              ;
              ;QUELLSEITENADRESSE BERECHNEN:
843F 20 D0 84       JSR  SEITTA  ;SEITENADRESSE TAUSCHEN
8442 20 12 85       JSR  WINIT   ;FENSTER-PARAMETER, INITIALISIERUNG
8445 20 D0 84       JSR  SEITTA  ;SEITENADRESSE TAUSCHEN
8448 90 03          BCC  GC04    ;KEIN VOLLES FENSTER
              ;
844A 4C DF 84       JMP  GCOV    ;ROUTINE FUER VOLLES FENSTER
              ;
844D 24 AE    GC04  BIT  GCFLAG
844F 10 03          BPL  GC00    ;KEINE ZIELKOORDINATEN ANGEGEBEN
8451 20 E0 85       JSR  WINIT2  ;ZIELKOORDINATEN INIT
              ;
8454 78       GC00  SEI
8455 A5 01          LDA  1
8457 48             PHA
8458 29 FC          AND  #$FC
845A 85 01          STA  1       ;ROM AUS
              ;
845C A0 00          LDY  #0
              ;
              ;
```

```
              ;GCOMB FUER FENSTER AUSFUEHREN:
              ;*******************************
              ;
845E 24 AE    GCO2     BIT  GCFLAG
8460 10 03             BPL  GCO3    ;KEIN ZIELFELD ANGEGEBEN
              ;
8462 20 D8 86          JSR  ZIELAD  ;ZIELADRESSE, MASKE HOLEN
              ;
8465 20 57 86 GCO3     JSR  GRIGHT  ;MASKE+QUELLADRESSE HOLEN
8468 90 3B             BCC  GCOEND  ;FERTIG
              ;
846A 24 AE             BIT  GCFLAG
846C 30 25             BMI  GCOJ    ;->ZIELFELD ANGEGEBEN
              ;
              ;IN GRIGHT BERECHNETE ADRESSE->ZIELADRESSE:
              ;
846E 48                PHA
846F A5 61             LDA  GRAD1H  ;QUELLADRESSE
8471 29 1F             AND  #$1F
8473 0D 4F C6          ORA  GRMEM   ;ADRESSE ERSTSEITE
8476 85 6A             STA  GRAD2H  ;NACH ZIELSPEICHER
8478 A5 60             LDA  GRAD1L
847A 85 69             STA  GRAD2L  ;LOW-BYTE
              ;
847C A5 62             LDA  VRAD1L
847E 85 6B             STA  VRAD2L
8480 85 6D             STA  FRAD2L  ;FARBSPEICHERADRESSEN
8482 A5 63             LDA  VRAD1H
8484 29 03             AND  #$03
8486 0D 50 C6          ORA  COLMEM  ;VIDEORAMADRESSE
8489 85 6C             STA  VRAD2H
848B 29 03             AND  #$03
848D 0D 51 C6          ORA  FABMEM  ;FARBRAMADRESSE
8490 85 6E             STA  FRAD2H
              ;
8492 68                PLA
              ;
              ;
              ;PROGRAMMMODIFIKATION:
8493 20 FF FF GCOJ     JSR  $FFFF   ;UNTERPROGRAMMSPRUNG ZUR COMB-ROUTINE
              ;
8496 A5 97             LDA  FLG
8498 D0 C4             BNE  GCO2    ;FARBE KOMMT AUS ZIELFELD
              ;
```

```
               ;FARBE AUS QUELLFELD:
               ;
849A B1 62             LDA  (VRAD1L),Y
849C 91 6B             STA  (VRAD2L),Y  ;VIDEORAM
849E B1 64             LDA  (FRAD1L),Y
84A0 91 6D             STA  (FRAD2L),Y  ;FARBRAM
84A2 4C 5E 84          JMP  GCO2        ;WEITER
               ;
               ;
               ;ENDE:
               ;*****
               ;
84A5 4C A5 83 GCOEND   JMP  GC4     ;BEENDEN
               ;
               ;
               ;
               ;INDIREKTE ROUTINEN
               ;FUER GCOMB (MASKE IN A):
               ;************************
               ;
               ;UND-VERKNUEPFUNG:
               ;
84A8 49 FF    GCOAND   EOR  #$FF        ;MASKE UMKEHREN
84AA 11 60             ORA  (GRAD1L),Y
84AC 31 69             AND  (GRAD2L),Y
84AE 91 69             STA  (GRAD2L),Y  ;NACH ZIELSPEICHER
84B0 60                RTS
               ;
               ;
               ;ODER-VERKNUEPFUNG:
               ;
84B1 31 60    GCOOR    AND  (GRAD1L),Y
84B3 11 69             ORA  (GRAD2L),Y
84B5 91 69             STA  (GRAD2L),Y
84B7 60                RTS
               ;
               ;
               ;EXODER-VERKNUEPFUNG:
               ;
84B8 31 60    GCOEOR   AND  (GRAD1L),Y
84BA 51 69             EOR  (GRAD2L),Y
84BC 91 69             STA  (GRAD2L),Y
84BE 60                RTS
               ;
               ;
```

```
               ;COPIEREN:
               ;
84BF 48        GCOCOP   PHA          ;MASKE MERKEN
84C0 31 60              AND  (GRAD1L),Y
84C2 8D 48 C6           STA  ZWIS
84C5 68                 PLA
84C6 49 FF              EOR  #$FF    ;MASKE UMDREHEN
84C8 31 69              AND  (GRAD2L),Y
84CA 0D 48 C6           ORA  ZWIS
84CD 91 69              STA  (GRAD2L),Y
84CF 60                 RTS
               ;
               ;
               ;GRAPHIKSEITENADRESSE TAUSCHEN:
               ;********************************
               ;
84D0 AD 4F C6 SEITTA    LDA  GRMEM
84D3 48                 PHA
84D4 AD 52 C6           LDA  GRMM2   ;BASISADRESSE SEITE 2
84D7 8D 4F C6           STA  GRMEM   ;NACH 1
84DA 68                 PLA
84DB 8D 52 C6           STA  GRMM2   ;SEITE 1 NACH 2
84DE 60                 RTS
               ;
               ;
               ;GCOMB FUER VOLLES GRAPHIKFENSTER:
               ;***********************************
               ;
84DF 78        GCOV     SEI
84E0 A5 01              LDA  1
84E2 48                 PHA
84E3 29 FC              AND  #$FC
84E5 85 01              STA  1       ;ROM AUS
               ;
84E7 20 1D 7D           JSR  STUGR   ;ADR. ZIELSEITE (BEFEHLIGTE SEITE) NACH USE
84EA AD 52 C6           LDA  GRMM2
84ED 84 69              STY  GRAD2L  ;Y=0!
84EF 85 6A              STA  GRAD2H  ;ADR. PARTNERSEITE
84F1 A2 20              LDX  #$20
               ;
               ;
84F3 B1 FD     GCO5     LDA  (USE),Y
               ;
```

```
                ;MODIFIZIERTER BEFEHL:
84F5 31 69      GCO6      AND  (GRAD2L),Y  ;VERKNUEPFEN
                ;
84F7 91 FD                STA  (USE),Y     ;NACH ZIELSEITE
84F9 C8                   INY
84FA D0 F7                BNE  GCO5
84FC E6 FE                INC  USE+1
84FE E6 6A                INC  GRAD2H
8500 CA                   DEX          ;ZAEHLER
8501 D0 F0                BNE  GCO5
                ;
8503 F0 A0                BEQ  GCOEND  ;ENDE
                ;
                ;
                ;
                ;BEFEHLSCODETABELLE FUER
                ;SELBSTMODIFIZIERENDES PROGRAMM:
                ;********************************
                ;
8505 31 11 51 COMBTB   .BYT $31,$11,$51,$B1
                ;AND/ORA/EOR/LDA (COPIE)
                ;
                ;
                ;
                ;SPRUNGADRESSENTABELLE FUER
                ;SELBSTMODIFIZIERENDES PROGRAMM:
                ;********************************
                ;
8509 A8 84      COMBT2    .WOR GCOAND
850B B1 84                .WOR GCOOR
850D B8 84                .WOR GCOEOR
850F BF 84                .WOR GCOCOP
                ;
                ;
                ;
                ;
                ;
                ;INITIALISIERUNGSROUTINEN:
                ;*************************
                ;
                ;FUER GRIGHT:
                ;************
                ;
8511 60         WINOO     RTS
                ;
8512 20 79 00 WINIT       JSR  CHRGOT
8515 38                   SEC          ;FLAG FUER VOLLES FENSTER
8516 F0 F9                BEQ  WINOO   ;KEINE FENSTER-PARAMETER->VOLLES FENSTER
                ;
```

```
8518 20 73 00           JSR  CHRGET  ;NAECHSTES ZEICHEN
851B A5 02              LDA  GFLAG
851D 48                 PHA
851E 29 FE              AND  #$FE
8520 85 02              STA  GFLAG   ;LGR-FLAG AUSBLENDEN
8522 20 58 81           JSR  FINITO  ;PARAMETER HOLEN->X2,Y2/X3,Y3
               ;
8525 A2 00              LDX  #0      ;FLAG
8527 20 79 00           JSR  CHRGOT  ;LETZTES ZEICHEN HOLEN
852A F0 25              BEQ  WIN4    ;KEINE ZIELKOORDINATEN ANGEGEBEN
852C C9 A4              CMP  #TO
852E F0 03              BEQ  WIN5
8530 20 08 AF           JSR  SERR    ;SYNTAX ERROR
8533 20 73 00 WIN5      JSR  CHRGET  ;NAECHSTES ZEICHEN HOLEN
8536 20 FF 7F           JSR  TESCOR  ;ZIELKOORDINATEN HOLEN
8539 29 F8              AND  #$F8    ;UNTERE 3 BITS AUSBLENDEN
853B 8D 3E C6           STA  X1L     ;SPEICHERN
853E AD 42 C6           LDA  X2L
8541 29 07              AND  #$07    ;DAFUER 3 BITS VON X2
8543 0D 3E C6           ORA  X1L     ;X-KOORD. IN GLEICHEN BLOCK
8546 8D 3E C6           STA  X1L     ;WIE QUELLFENSTER BRINGEN
8549 8E 3F C6           STX  X1H
854C 8C 3C C6           STY  Y1      ;MERKEN
854F A2 80              LDX  #$80    ;FLAG
               ;
8551 86 AE     WIN4     STX  GCFLAG  ;FLAG MERKEN
8553 68                 PLA
8554 85 02              STA  GFLAG   ;GFLAG WIEDERHERSTELLEN
               ;
8556 AD 46 C6           LDA  X3L
8559 AE 47 C6           LDX  X3H
855C AC 44 C6           LDY  Y3
855F 85 5A              STA  WX2L
8561 86 5B              STX  WX2H
8563 84 5C              STY  WY2     ;KOORD. UNTERE RECHTE ECKE
               ;
               ;VOLLES FENSTER?
               ;
               ;UNTERE RECHTE KOORDINATEN TESTEN:
8565 C0 C7              CPY  #199
8567 D0 21              BNE  WIN1    ;NEIN!
8569 24 02              BIT  GFLAG
856B 30 0A              BMI  WIN2    ;MC
               ;HGR:
856D E0 3F              CPX  #<319
856F D0 19              BNE  WIN1    ;NEIN!
8571 C9 01              CMP  #>319
8573 D0 15              BNE  WIN1    ;NEIN!
8575 F0 08              BEQ  WIN3    ;C=1!/VIELLEICHT!
               ;MC:
```

```
8577 E0 9F    WIN2     CPX  #<159
8579 D0 0F             BNE  WIN1     ;NEIN!
857B C9 00             CMP  #>159
857D D0 0B             BNE  WIN1     ;NEIN!
              ;
              ;OBERE LINKE KOORDINATE TESTEN:
857F AD 42 C6 WIN3     LDA  X2L
8582 0D 43 C6          ORA  X2H
8585 0D 40 C6          ORA  Y2       ;KOORDINATEN=0,0?
8588 F0 55             BEQ  WIN0     ;JA!->VOLLES FENSTER! (C=1!)
              ;
              ;WEITER:
              ;
858A AD 42 C6 WIN1     LDA  X2L
858D AE 43 C6          LDX  X2H
8590 AC 40 C6          LDY  Y2
              ;
8593 85 57    WINIT0   STA  WX1L
8595 86 58             STX  WX1H
8597 84 59             STY  WY1      ;KOORD. OBERE LINKE ECKE
8599 85 5D             STA  WXAL
859B 86 5E             STX  WXAH
859D 84 5F             STY  WYA      ;NACH AKTUELLEN KOORDINATEN
              ;
859F 20 AA 81          JSR  HPOSN    ;PUNKTADRESSE BERECHNEN
              ;
              ;ADRESSEN DECREMIEREN
85A2
              ;
85A2 A5 AC    WINITS   LDA  ADL      ;ZEILENSTARTADRESSE NACH GRAD1:
85A4 85 62             STA  VRAD1L
85A6 38                SEC
85A7 E9 08             SBC  #8
85A9 85 60             STA  GRAD1L   ;GRAPHIKADRESSE 8 ZURUECK
85AB A5 AD             LDA  ADH
85AD 48                PHA
85AE E9 00             SBC  #0
85B0 85 61             STA  GRAD1H
              ;
85B2 68                PLA
85B3 4A                LSR  A
85B4 66 62             ROR  VRAD1L
85B6 4A                LSR  A
85B7 66 62             ROR  VRAD1L
85B9 4A                LSR  A
85BA 66 62             ROR  VRAD1L       ;GRAPHIKADRESSE/8
85BC 49 DC             EOR  #%11011100 ;BASISADRESSE VIDEORAM
85BE 85 63             STA  VRAD1H       ;NACH VIDEORAMADRESSE
              ;
```

```
85C0 38                 SEC
85C1 A5 62              LDA  VRAD1L
85C3 E9 01              SBC  #1
85C5 85 62              STA  VRAD1L     ;VIDEORAMADRESSE-1
85C7 85 64              STA  FRAD1L
85C9 B0 02              BCS  WIN8
85CB C6 63              DEC  VRAD1H
                ;
85CD A5 63      WIN8    LDA  VRAD1H
85CF 29 03              AND  #3
85D1 09 40              ORA  #%01000000 ;6.BIT SETZEN
85D3 24 61              BIT  GRAD1H     ;6. BIT GRAPHIKADRESSE=1?
85D5 70 03              BVS  WIN9    ;JA->$E000-$FFFF
85D7 09 CC              ORA  #$CC    ;NEIN+>$A000-$BFFF//ZWEITFARBRAM
85D9 2C                 .BYT $2C     ;NAECHSTEN BEFEHL UEBERSPRINGEN
85DA 09 D8      WIN9    ORA  #$D8    ;AKTUELLER FARBRAM
85DC 85 65              STA  FRAD1H  ;FARBRAMADRESSE (HIGH)
                ;
85DE 18                 CLC          ;FLAG
85DF 60         WIN0    RTS
                ;
                ;
                ;
                ;INITIALISIERUNG FUER ZIELFELD (GCOMB):
                ;****************************************
                ;
85E0 38         WINIT2  SEC          ;GROESSE DES FELDES BERECHNEN:
85E1 A5 5A              LDA  WX2L
85E3 E5 57              SBC  WX1L
85E5 AA                 TAX
85E6 A5 5B              LDA  WX2H
85E8 E5 58              SBC  WX1H
85EA 48                 PHA
85EB 8A                 TXA
85EC 48                 PHA          ;X2-X1 AUF STACK
85ED 38                 SEC
85EE A5 5C              LDA  WY2
85F0 E5 59              SBC  WY1
85F2 48                 PHA          ;Y2-Y1 AUF STACK
                ;
85F3 20 20 86           JSR  TAUSCH  ;REGISTERSATZ 1 RETTEN
                ;
                ;UNTERE RECHTE ECKE VON ZIELFELD BERECHNEN:
                ;
85F6 18                 CLC
85F7 68                 PLA
85F8 6D 3C C6           ADC  Y1
85FB 85 5C              STA  WY2     ;Y-KOORDINATE
85FD B0 1E              BCS  WERROR  ;AUSSERHALB BILDSCHIRM
85FF 18                 CLC
8600 68                 PLA
```

```
8601 6D 3E C6          ADC  X1L
8604 85 5A             STA  WX2L
8606 68                PLA
8607 6D 3F C6          ADC  X1H
860A 85 5B             STA  WX2H    ;X-KOORDINATE
860C B0 0F             BCS  WERROR  ;AUSSERHALB BILDSCHIRM
              ;
860E AD 3E C6          LDA  X1L
8611 AE 3F C6          LDX  X1H
8614 AC 3C C6          LDY  Y1
8617 20 93 85          JSR  WINITO  ;PUNKTADRESSE BERECHNEN
861A 4C 20 86          JMP  TAUSCH  ;WIEDER WECHSELN
              ;
861D 4C 48 B2 WERROR   JMP  QERR    ;ILLEGAL QUANTITY
              ;
              ;
              ;REGISTERSATZ 1 UND 2 AUSTAUSCHEN:
              ;**********************************
              ;
8620 A2 0E    TAUSCH   LDX  #14
8622 B5 57    TAUO     LDA  WX1L,X  ;AKTIVES REGISTER AUS NULLSEITE
8624 48                PHA
8625 BD 46 86          LDA  SPRET,X ;GERETTETES REGISTER
8628 95 57             STA  WX1L,X  ;NACH NULLSEITE
862A 68                PLA
862B 9D 46 86          STA  SPRET,X ;REGISTER RETTEN
862E CA                DEX
862F 10 F1             BPL  TAUO
8631 A5 AC             LDA  ADL
8633 AE 55 86          LDX  SPRET+15
8636 8D 55 86          STA  SPRET+15
8639 86 AC             STX  ADL
863B A5 AD             LDA  ADH
863D AE 56 86          LDX  SPRET+16
8640 8D 56 86          STA  SPRET+16
8643 86 AD             STX  ADH     ;ADL/ADH AUSTAUSCHEN
8645 60                RTS
              ;
              ;
              ;SPEICHER FUER 17 GERETTETE REGISTER:
              ;*************************************
              ;(2. REGISTERSATZ)
              ;
8646 00 00 00 SPRET    .BYT 0,0,0,0,0
864B 00 00 00          .BYT 0,0,0,0,0
8650 00 00 00          .BYT 0,0,0,0,0
8655 00 00             .BYT 0,0     ;FUER ADL/ADH
              ;
              ;
              ;
              ;
```

```
              ;
              ;ERMITTELN DER ADRESSE UND MASKE
              ;DES NAECHSTEN BYTES EINES FENSTER-FELDES:
              ;*****************************************
              ;
              ;EINGABE:
              ;         WXAL/H:AKTUELLE KOORDINATEN
              ;         WYA   :DES STARTBITS DES GESUCHTEN BYTES
              ;         GRAD1L/
              ;         GRAD1H:ADRESSE DES VORHERIGEN BYTES
              ;         VRAD1L/
              ;         GRAD1H:ADRESSE DES VORHERIGEN VIDEORAMBYTES
              ;         FRAD1L/
              ;         FRAD1H:ADRESSE DES VORHERIGEN FARBRAMBYTES
              ;         WX1L/H:FENSTER-
              ;         WY1   :KOORDINATEN OBEN LINKS
              ;         WX2L/H:FENSTER-
              ;         WY2   :KOORDINATEN UNTEN RECHTS
              ;         ADL/
              ;         ADH   :ADRESSE DES ZEILENSTARTS
              ;
              ;AUSGABE:
              ;          ALLE OBIGEN REGISTER
              ;          AKTUALISIERT FUER NAECHSTES BYTE
              ;
              ;          AKU  :MASKE:
              ;                ALLE BITS DES BYTES,
              ;                DIE ZUM FENSTER GEHOEREN
              ;                SIND GLEICH 1
              ;          CARRY:=0: FENSTER FERTIG
              ;                =1: FENSTER NOCH NICHT FERTIG
              ;
              ;
              ;
8657 A5 5D    GRIGHT   LDA  WXAL     ;AKTUELLE X-KOORD. LOW
8659 29 07             AND  #$07     ;BERECHNE BITPOSITION DES PUNKTES IM
SPEICHER
865B 8D F9 86          STA  BIPO     ;ZWISCHENSPEICHERN
865E AA                TAX             ;=OFFSET ZU TABELLE
865F BD C8 86          LDA  BIPTB1,X  ;HOLE MASKE
8662 48                PHA             ;MERKEN
              ;
8663 38                SEC
8664 A5 5A             LDA  WX2L
8666 E5 5D             SBC  WXAL     ;X2-XA
8668 AA                TAX
8669 A5 5B             LDA  WX2H     ;NOCH EIN VOLLES BYTE
866B E5 5E             SBC  WXAH     ;BIS X2?
866D 90 36             BCC  GRI0     ;<0 => NAECHSTE REIHE
866F D0 0E             BNE  GRI1     ;>=8 -> JA, VOLLES BYTE
8671 E0 08             CPX  #8
```

```
8673 B0 0A             BCS  GRI1     ;>=8 -> JA
              ;
              ;ENDE DES FENSTERS ERREICHT->MASKE KUERZEN:
              ;
8675 A5 5A             LDA  WX2L
8677 29 07             AND  #$07
8679 AA                TAX            ;OFFSET FUER ZWEITE TABELLE
867A 68                PLA
867B 3D D0 86          AND  BIPTB2,X ;MASKE HINTEN KUERZEN
867E 48                PHA            ;UND MERKEN
              ;
867F AD F9 86 GRI1     LDA  BIPO
8682 49 07             EOR  #$07     ;7-BIPO->AKU
8684 38                SEC           ;C=1!!!
8685 65 5D             ADC  WXAL
8687 85 5D             STA  WXAL     ;NAECHSTER BYTE-START
8689 90 02             BCC  GRI2
868B E6 5E             INC  WXAH     ;8-BIPO + XA -> XA
              ;
868D A5 60    GRI2     LDA  GRAD1L
868F 18                CLC
8690 69 08             ADC  #8
8692 85 60             STA  GRAD1L   ;ADRESSE ZUM NAECHST-RECHTEN BYTE
8694 90 02             BCC  GRI3
8696 E6 61             INC  GRAD1H
              ;
8698 E6 64    GRI3     INC  FRAD1L
869A E6 62             INC  VRAD1L
869C D0 04             BNE  GRI4
869E E6 65             INC  FRAD1H   ;FARBRAMADRESSE+1
86A0 E6 63             INC  VRAD1H   ;VIDEORAMADRESSE+1
              ;
86A2 68       GRI4     PLA           ;MASKE HOLEN
86A3 38                SEC           ;FLAG FUER NICHT FERTIG
86A4 60       GRI5     RTS           ;ZURUECK
              ;
              ;
              ;NAECHSTE REIHE:
              ;
86A5 68       GRI0     PLA
86A6 E6 5F             INC  WYA      ;YA+1
86A8 A5 5C             LDA  WY2
86AA C5 5F             CMP  WYA      ;Y2<YA?
86AC 90 F6             BCC  GRI5     ;JA->FENSTER FERTIG C=0->FERTIG-FLAG
              ;
86AE A5 AD             LDA  ADH
86B0 29 E0             AND  #%11100000
86B2 69 1D             ADC  #$1D     ;C=1!
86B4 8D 17 C6          STA  GEND     ;GEND FUER UNTEN
86B7 A5 57             LDA  WX1L
86B9 85 5D             STA  WXAL
```

```
86BB A5 58              LDA  WX1H    ;X1->XA (SPALTENANFANG)
86BD 85 5E              STA  WXAH    ;= STARTPUNKT NAECHSTE ZEILE
86BF 20 FA 86           JSR  UNTEN   ;EINE ZEILE TIEFER (ADR. IN ADL/ADH)
86C2 20 A2 85           JSR  WINITS  ;INIT-ADRESSEN BERECHNEN
86C5 4C 57 86           JMP  GRIGHT  ;WEITER MACHEN
               ;
               ;
               ;
               ;TABELLEN DER BITPOSITIONSMASKEN:
               ;********************************
               ;
86C8 FF        BIPTB1   .BYT %11111111
86C9 7F                 .BYT %01111111
86CA 3F                 .BYT %00111111
86CB 1F                 .BYT %00011111
86CC 0F                 .BYT %00001111
86CD 07                 .BYT %00000111
86CE 03                 .BYT %00000011
86CF 01                 .BYT %00000001
               ;
86D0 80        BIPTB2   .BYT %10000000
86D1 C0                 .BYT %11000000
86D2 E0                 .BYT %11100000
86D3 F0                 .BYT %11110000
86D4 F8                 .BYT %11111000
86D5 FC                 .BYT %11111100
86D6 FE                 .BYT %11111110
86D7 FF                 .BYT %11111111
               ;
               ;
               ;
               ;DIE GLEICHE ROUTINE MIT VORHERIGEM
               ;AUSTAUSCH DER REGISTER ZUR BERECHNUNG
               ;EINES ZIELFENSTERS FUER GCOMB:
               ;*************************************
               ;
               ;
86D8 20 20 86 ZIELAD    JSR  TAUSCH  ;REGISTERSATZ TAUSCHEN
86DB 20 57 86           JSR  GRIGHT  ;NAECHSTES BYTE
86DE A5 60              LDA  GRAD1L
86E0 85 69              STA  GRAD2L
86E2 A5 61              LDA  GRAD1H
86E4 85 6A              STA  GRAD2H  ;GRAPHIKADRESSE MERKEN
86E6 A5 62              LDA  VRAD1L
86E8 85 6B              STA  VRAD2L
86EA A5 63              LDA  VRAD1H
86EC 85 6C              STA  VRAD2H  ;VIDEORAMADRESE MERKEN
86EE A5 64              LDA  FRAD1L
```

```
86F0 85 6D              STA  FRAD2L
86F2 A5 65              LDA  FRAD1H
86F4 85 6E              STA  FRAD2H  ;FARBRAMADRESSE MERKEN
86F6 4C 20 86           JMP  TAUSCH  ;REGISTERSATZ TAUSCHEN
               ;
               ;
               ;
               ;SPEICHER:
               ;*********
               ;
86F9 00        BIPO     .BYT 0
               ;
               ;
               ;
               ;
               ;VEKTORROUTINEN:
               ;***************
               ;
               ;
               ;GRAPHIKCURSOR NACH UNTEN:
               ;
86FA 24 02     UNTEN    BIT  GFLAG   ;GRAPHIKFLAG
86FC 70 76              BVS  JUNTEN  ;LGR/SPRITE-MODUS
               ;
86FE A5 AC              LDA  ADL     ;PUNKTADRESSE (LOW-BYTE)
8700 29 07              AND  #7
8702 C9 07              CMP  #7
8704 F0 05              BEQ  UN1     ;PACKENRAND
8706 38                 SEC
8707 A9 00              LDA  #0
8709 B0 1D              BCS  UN3     ;ADDIERE 1 (C=1!)
               ;
870B A5 AD     UN1      LDA  ADH
870D CD 17 C6           CMP  GEND
8710 90 12              BCC  UN2
8712 A5 AC              LDA  ADL
8714 E9 07              SBC  #7
8716 85 AC              STA  ADL
8718 A5 AD              LDA  ADH
871A E9 1E              SBC  #$1E
871C 85 AD              STA  ADH
871E A9 01              LDA  #1
8720 8D 91 C6           STA  RUECK   ;RUECKMELDUNG F. AUSSERHALB FENSTER
8723 60                 RTS
               ;
```

```
8724 A9 39     UN2      LDA  #$39     ;ADDIERE 320-7=313
8726 E6 AD              INC  ADH      ;C=0!
8728 65 AC     UN3      ADC  ADL
872A 85 AC              STA  ADL
872C A9 00              LDA  #0
872E 65 AD              ADC  ADH
8730 85 AD              STA  ADH
8732 60                 RTS
               ;
               ;
               ;OBEN/UNTEN-ENTSCHEIDUNG:
               ;
8733 30 C5     U.O      BMI  UNTEN
               ;
               ;
               ;GRAPHIKCURSOR NACH OBEN:
               ;
8735 24 02     OBEN     BIT  GFLAG
8737 70 3E              BVS  JOBEN    ;LGR/SPRITE-MODUS
               ;
8739 A5 AC              LDA  ADL
873B 29 07              AND  #7
873D F0 05              BEQ  OB1      ;PACKENRAND (OBEN)
873F 18                 CLC
8740 A9 FF              LDA  #$FF
8742 90 25              BCC  OB4      ;SUBTRAHIERE 1
               ;
8744 AD 18 C6  OB1      LDA  GSTART
8747 C5 AD              CMP  ADH
8749 90 1A              BCC  OB3
874B D0 06              BNE  OB2
874D A9 3F              LDA  #$3F
874F C5 AC              CMP  ADL
8751 90 12              BCC  OB3
               ;
8753 A5 AC     OB2      LDA  ADL
8755 69 06              ADC  #6
8757 85 AC              STA  ADL
8759 A5 AD              LDA  ADH
875B 69 1E              ADC  #$1E
875D 85 AD              STA  ADH
875F A9 02              LDA  #2
8761 8D 91 C6           STA  RUECK    ;AUSSERHALB FENSTER
8764 60                 RTS
               ;
```

```
8765 A9 C7    OB3     LDA  #$C7    ;SUBTRAHIERE 320+7=327
8767 C6 AD            DEC  ADH     ;C=0!
8769 65 AC    OB4     ADC  ADL
876B 85 AC            STA  ADL
876D A5 AD            LDA  ADH
876F E9 00            SBC  #0
8771 85 AD            STA  ADH
8773 60               RTS
              ;
              ;
              ;SPRUENGE ZU DEN LGR/SPRITE-VEKTORROUTINEN:
              ;
8774 4C 62 7F JUNTEN  JMP  SUNTEN
8777 4C 6C 7F JOBEN   JMP  SOBEN
877A 4C 76 7F JRECHT  JMP  SRECHT
877D 4C 94 7F JLINKS  JMP  SLINKS
              ;
              ;
              ;GRAPHIKCURSOR NACH RECHTS:
              ;
8780 18       RECHTS  CLC
8781 24 02            BIT  GFLAG
8783 70 F5            BVS  JRECHT  ;LGR/SPRITE-MODUS
              ;
8785 10 02            BPL  RE1
8787 66 AB            ROR  MSK     ;MC
8789 66 AB    RE1     ROR  MSK     ;BITMASKE ROLLEN
878B 90 1B            BCC  RE2     ;NOCH INNERHALB BYTE
878D 66 AB            ROR  MSK     ;AUSSERHALB BYTE
878F A5 AC            LDA  ADL
8791 C8               INY
8792 C0 28            CPY  #$28
8794 38               SEC
8795 D0 12            BNE  RE3
8797 A0 00            LDY  #0
8799 E9 38            SBC  #$38
879B 85 AC            STA  ADL
879D A5 AD            LDA  ADH
879F E9 01            SBC  #1
87A1 85 AD            STA  ADH
87A3 A9 03            LDA  #3
87A5 8D 91 C6         STA  RUECK   ;AUSSERHALB FENSTER
87A8 60       RE2     RTS
              ;
87A9 69 07    RE3     ADC  #7
87AB 85 AC            STA  ADL
87AD 90 F9            BCC  RE2
87AF E6 AD            INC  ADH     ;8 ADDIEREN
87B1 60               RTS
```

```
                ;
                ;
                ;RECHTS/LINKS-ENTSCHEIDUNG:
                ;
87B2 10 CC      R.L       BPL  RECHTS
                ;
                ;
                ;GRAPHIKCURSOR NACH LINKS:
                ;
87B4 18         LINKS     CLC
87B5 24 02                BIT  GFLAG
87B7 70 C4                BVS  JLINKS  ;LGR/SPRITE-MODUS
87B9 10 02                BPL  LI1
87BB 26 AB                ROL  MSK     ;MC
87BD 26 AB      LI1       ROL  MSK
87BF 90 19                BCC  LI2
87C1 26 AB                ROL  MSK     ;BITMASKE ROLLEN
87C3 A5 AC                LDA  ADL
87C5 88                   DEY
87C6 18                   CLC
87C7 10 12                BPL  LI3
87C9 A0 27                LDY  #$27
87CB 69 38                ADC  #$38
87CD 85 AC                STA  ADL
87CF A5 AD                LDA  ADH
87D1 69 01                ADC  #1
87D3 85 AD                STA  ADH
87D5 A9 04                LDA  #4
87D7 8D 91 C6             STA  RUECK   ;AUSSERHALB FENSTER
87DA 60         LI2       RTS
                ;
87DB E9 07      LI3       SBC  #7
87DD 85 AC                STA  ADL
87DF B0 F9                BCS  LI2
87E1 C6 AD                DEC  ADH     ;8 SUBTRAHIEREN
87E3 60                   RTS
                ;
                ;
                ;
```

```
               ;LINIE VON X1,Y1 NACH X2,Y2 ZEICHNEN:
               ;(ZENTRALROUTINE)
               ;*************************************
               ;
               ;EINGANG:
               ;          A: X2-KOORDINATE (LOW-BYTE)
               ;          X: X2-KOORDINATE (HIGH-BYTE)
               ;          Y: Y2-KOORDINATE
               ;         E0: X1-KOORDINATE (LOW-BYTE)
               ;         E1: X1-KOORDINATE (HIGH-BYTE)
               ;         E2: Y1-KOORDINATE
               ;
               ;
87E4 48        HLINE    PHA           ;A  : X2-LOW-BYTE RETTEN
87E5 20 63 88           JSR  GETE5    ;E5 BERECHNEN
87E8 68                 PLA
87E9 48                 PHA           ;X2-K-L
87EA 38                 SEC
87EB ED 10 C6           SBC  E0
87EE 48                 PHA
87EF 8A                 TXA
87F0 ED 11 C6           SBC  E1
87F3 85 5A              STA  D3       ;X2-X1
87F5 B0 0A              BCS  L4       ;NEGATIV?
87F7 68                 PLA           ;JA
87F8 49 FF              EOR  #$FF
87FA 69 01              ADC  #1
87FC 48                 PHA
87FD A9 00              LDA  #0
87FF E5 5A              SBC  D3       ;VZW
8801 85 58     L4       STA  D1
8803 85 5C              STA  D5       ;(X2-X1)-HIGH
8805 68                 PLA
8806 85 57              STA  D0
8808 85 5B              STA  D4       ;(X2-X1)-LOW
880A 68                 PLA
880B 8D 10 C6           STA  E0
880E 8E 11 C6           STX  E1       ;X2 ALS NEUE X1
               ;
8811 98                 TYA
8812 18                 CLC
8813 ED 12 C6           SBC  E2       ;Y2-Y1
8816 90 04              BCC  HL4      ;NEGATIV
8818 49 FF              EOR  #$FF
881A 69 FE              ADC  #$FE     ;VZW
881C 85 59     HL4      STA  D2       ;(Y2-Y1)
881E 8C 12 C6           STY  E2       ;Y2 ALS NEUE Y1
               ;
```

```
8821 66 5A              ROR  D3      ;(X2-X1)/2
8823 38                 SEC
8824 E5 57              SBC  D0      ;(Y2-Y1)-(X2-X1)
8826 AA                 TAX          ;LOW-BYTE NACH X-REG.
8827 A9 FF              LDA  #$FF
8829 E5 58              SBC  D1
882B 85 5F              STA  ZA      ;HIGH-BYTE NACH ZA
882D A4 93              LDY  E5
882F B0 05              BCS  L5      ;UNBEDINGT
                 ;
                 ;ZEICHENSCHLEIFE:
                 ;
8831 0A          L1     ASL  A
8832 20 B2 87           JSR  R.L     ;RECHTS/LINKS
8835 38                 SEC
8836 A5 5B       L5     LDA  D4
8838 65 59              ADC  D2
883A 85 5B              STA  D4
883C A5 5C              LDA  D5
883E E9 00              SBC  #0      ;(X2-X1)-(Y2-Y1) -> (X2-X1)
8840 85 5C       L2     STA  D5
                 ;
8842 84 93              STY  E5
8844 20 18 82           JSR  PLT     ;PUNKT ZEICHNEN
                 ;
8847 E8          L1.    INX          ;EINSPRUNG FUER SPRITE-IRQ
8848 D0 05              BNE  L6
884A E6 5F              INC  ZA
884C D0 01              BNE  L6      ;ZAEHLER
884E 60                 RTS
                 ;
884F A5 5A       L6     LDA  D3
8851 B0 DE              BCS  L1
8853 20 33 87           JSR  U.O     ;UNTEN/OBEN
8856 18                 CLC
8857 A5 5B              LDA  D4
8859 65 57              ADC  D0
885B 85 5B              STA  D4
885D A5 5C              LDA  D5
885F 65 58              ADC  D1
8861 50 DD              BVC  L2      ;UNBEDINGT
                 ;
                 ;
                 ;
```

```
              ;E5 HOLEN:
              ;*********
              ;
8863 AD 11 C6 GETE5    LDA  E1
8866 4A                LSR  A
8867 AD 10 C6          LDA  E0
886A 6A                ROR  A
886B 4A                LSR  A
886C 24 02             BIT  GFLAG   ;BEI MC NUR DURCH 4
886E 30 01             BMI  GETE5.
8870 4A                LSR  A
8871 85 93    GETE5.   STA  E5
8873 60                RTS
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: GMOVE  **
              ;**                 **
              ;*********************
              ;
              ;
8874 20 9E B7 GMOVE    JSR  GETBYT  ;HOLE MODUS NACH X
8877 8A                TXA
8878 4A                LSR  A       ;0. BIT NACH CARRY
8879 08                PHP          ;RE/LI-FLAG RETTEN (C=0- LI / C=1- RE)
887A 29 01             AND  #1      ;EVT. FEHLER KORRIGIEREN
887C 4A                LSR  A
887D 6A                ROR  A       ;7.BIT F. SCROLL/SCHIEB VORBEREITEN
887E 85 96             STA  FLG2    ;7.BIT=0: SCROLL / =1: SCHIEB
8880 20 F1 B7          JSR  CHKGET  ;AUF KOMMA PRUEFEN UND STARTZEILE HOLEN
8883 E0 19             CPX  #25
8885 90 02             BCC  GMV8
8887 A2 18             LDX  #24     ;KORREKTUR
8889 8E 3C C6 GMV8     STX  Y1      ;ALS OBERE ZEILE MERKEN (O)
888C 20 F1 B7          JSR  CHKGET  ;ENDZEILE HOLEN
888F E0 19             CPX  #25
8891 90 02             BCC  GMV7
8893 A2 18             LDX  #24     ;KORREKTUR
8895 8E 40 C6 GMV7     STX  Y2      ;ALS UNTERE ZEILE MERKEN (U)
8898 8A                TXA          ;O NACH A
              ;
```

```
               ;O UND U NACH GROESSE ORDNEN:
               ;
8899 AE 3C C6           LDX  Y1       ;U NACH X
889C AC 40 C6           LDY  Y2       ;O NACH Y
889F 38                 SEC
88A0 ED 3C C6           SBC  Y1       ;O-U
88A3 B0 08              BCS  GMV1     ;O>=U => OK.
88A5 49 FF              EOR  #$FF     ;O<U
88A7 AE 40 C6           LDX  Y2       ;WECHSELN
88AA AC 3C C6           LDY  Y1
               ;
               ;RECHTS/LINKS-INITIALISIERUNG:
               ;
88AD 85 97     GMV1     STA  FLG      ;ZEILENZAEHLER
88AF 28                 PLP           ;RE/LI-FLAG HOLEN
88B0 08                 PHP           ;UND MERKEN
88B1 90 03              BCC  GMV0     ;->LINKSSCHIEBUNG
88B3 C8                 INY           ;RECHTS: 1 ZEILE WEITER
88B4 98                 TYA
88B5 AA                 TAX           ;UND UNTEN STARTEN
               ;
               ;HGR-STARTADRESSE BERECHNEN:
               ;
88B6 BD 37 7D  GMV0     LDA  MUL.H,X ;ZEILE MAL 320 (MULTIPLIKATIONSTABELLE)
88B9 0D 4F C6           ORA  GRMEM    ;AB AKTUELLER GRAPHIKSEITE
88BC 85 FE              STA  USE+1    ;ZEILENSTARTADRESSE HIGH-BYTE
88BE 8A                 TXA
88BF 29 03              AND  #3
88C1 AA                 TAX
88C2 28                 PLP
88C3 78                 SEI
88C4 A5 01              LDA  1
88C6 48                 PHA
88C7 29 FC              AND  #$FC
88C9 85 01              STA  1        ;ROM AUS
88CB 08                 PHP           ;RE/LI-FLAG HOLEN
88CC BD 51 7D           LDA  MUL.L,X ;LOW-BYTE DER ADRESSE
88CF 90 06              BCC  GMV9     ;LINKS
88D1 E9 08              SBC  #8
88D3 B0 02              BCS  GMV9
88D5 C6 FE              DEC  USE+1    ;-8 BEI RE(ENDE ZEILE)
88D7 85 FD     GMV9     STA  USE      ;NACH USE
88D9 85 61              STA  $61      ;FARBSPEICHERADRESSE:
88DB A5 FE              LDA  USE+1
88DD 4A                 LSR  A
88DE 66 61              ROR  $61
88E0 4A                 LSR  A
88E1 66 61              ROR  $61
88E3 4A                 LSR  A
88E4 66 61              ROR  $61      ;/8
88E6 29 03              AND  #3
```

```
88E8 0D 50 C6          ORA   COLMEM
88EB 85 62             STA   $62      ;COLORADRESSE NACH $61/$62
88ED 8D 48 C6          STA   ZWIS
                ;
                ;HGR(MC) UND LGR - VERZWEIGUNG:
                ;
88F0 A5 02             LDA   GFLAG    ;GRAPHIKFLAG
88F2 4A                LSR   A
88F3 B0 3A             BCS   LGRMOV   ;LGR
88F5 AD 55 C6          LDA   COLOR
88F8 85 15             STA   XK+1     ;NACHSCHUB FUER VIDEORAM-SCHIEB
88FA A5 FD             LDA   USE      ;C=0!
88FC 69 08             ADC   #8
88FE 85 AE             STA   SHP
8900 A5 FE             LDA   USE+1
8902 69 00             ADC   #0
8904 85 AF             STA   SHP+1    ;STARTADRESSE PLUS 8 MERKEN
                ;
                ;
                ;HGR/MC-VERSCHIEBUNG:
                ;********************
                ;
                ;JEDE ZEILE MIT JE 320 BYTES WIRD
                ;IN 5 ABSCHNITTE ZU JE 64 BYTES UNTERTEILT
                ;(GESCHWINDIGKEITSOPTIMIERUNG)
                ;
                ;
                ;A. ZEILENSCHLEIFE:
                ;
8906 A0 07      GMV2   LDY   #7
8908 B1 FD      LRV1   LDA   (USE),Y ;HERUEBER ZU ROLLENDES BYTE
890A AA                TAX            ;NACH X
890B 24 96             BIT   FLG2
890D 10 02             BPL   GMV5     ;ROLLEN
890F A2 00             LDX   #0       ;SCHIEB => LEERSTELLEN NACHSCHIEBEN
8911 96 57      GMV5   STX   D0,Y     ;ERSTE 8 BYTES ZWISCHENSPEICHERN (FUER
ROLLEN)
8913 88                DEY
8914 10 F2             BPL   LRV1
                ;
8916 A2 05             LDX   #5       ;ABSCHNITTSZAEHLER
8918 28                PLP
8919 08                PHP            ;RE/LI-FLAG
891A B0 06             BCS   GMV3
891C 20 74 89          JSR   LIVER    ;ZEILE LINKS VERSCHIEBEN
891F 4C 25 89          JMP   GMV4
8922 20 E0 89   GMV3   JSR   REVER    ;ZEILE RECHTS VERSCHIEBEN
8925 C6 97      GMV4   DEC   FLG
8927 10 DD             BPL   GMV2     ;NAECHSTE ZEILE-------
                ;
                ;
```

```
              ;ENDE:
              ;
8929 28                PLP            ;RE/LI-FLAG VOM STACK
892A 68                PLA
892B 85 01             STA  1         ;ROM AN
892D 58                CLI
892E 60                RTS
              ;
              ;
              ;
              ;LGR-VERSCHIEBUNG:
              ;******************
              ;
892F 28       LGRMOV   PLP            ;RE/LI-FLAG VOM STACK
8930 68                PLA
8931 85 01             STA  1         ;ROM AN
8933 08                PHP            ;UND WIEDER DRAUF
              ;
              ;ZEILENSCHLEIFE:
              ;
8934 A9 20    LGMVO    LDA  #" "      ;LEERZEICHEN ALS
8936 85 15             STA  XK+1      ;NACHSCHUB F. SCHIEB
              ;
8938 A5 62             LDA  $62
893A 29 03             AND  #3
893C 09 04             ORA  #4        ;BASISADRESSE=$0400
              ;
893E 28                PLP
893F 08                PHP            ;FLAG
8940 20 6A 89          JSR  LGMV1     ;ZEILE VERSCHIEBEN
8943 28                PLP
8944 08                PHP            ;FLAG
8945 A5 61             LDA  $61
8947 90 0A             BCC  LGMV4     ;LINKS
              ;
              ;RECHTS:
8949 69 27             ADC  #39       ;C=1!
894B 85 61             STA  $61       ;ADRESSE NAECHSTE ZEILE
894D 90 0C             BCC  LGMV3
894F E6 62             INC  $62
8951 B0 08             BCS  LGMV3     ;UNBEDINGT
              ;
              ;LINKS:
8953 E9 27    LGMV4    SBC  #39       ;C=0!
8955 85 61             STA  $61
8957 B0 02             BCS  LGMV3
8959 C6 62             DEC  $62       ;ADRESSE NAECHSTE ZEILE
              ;
```

```
              ;FARBVERSCHIEBUNG:
              ;
895B AD 1D C6 LGMV3    LDA  TCOL    ;TEXTHINTERGRUND
895E 28                PLP
895F 08                PHP          ;RE/LI-FLAG
8960 20 53 8A          JSR  FABRSC  ;FARBRAM VERSCHIEBEN
              ;
              ;
8963 C6 97             DEC  FLG     ;ZAEHLER
8965 10 CD             BPL  LGMVO   ;NAECHSTE ZEILE
              ;
              ;ENDE:
              ;
8967 28                PLP
8968 58                CLI
8969 60                RTS
              ;
              ;
              ;LGR-ZEILE LINKSVERSCHIEBEN:
              ;****************************
              ;
896A 85 62    LGMV1    STA  $62
896C B0 03             BCS  LGMV2
896E 4C BC 89          JMP  LVC
              ;
              ;
              ;LGR-ZEILE RECHTSVERSCHIEBEN:
              ;*****************************
              ;
8971 4C 1D 8A LGMV2    JMP  RVC
              ;
              ;
              ;
              ;HGR/MC-ZEILE LINKSVERSCHIEBEN:
              ;*******************************
              ;
              ;
              ;5TEL-ABSCHNITTSSCHLEIFE:
              ;
8974 A0 3F    LIVER    LDY  #63
8976 B1 AE    LV1      LDA  (SHP),Y  ;64 BYTES HOLEN
8978 99 00 DC          STA  $DC00,Y  ;RAM!-ZWISCHENSPEICHERN
897B 88                DEY
897C 10 F8             BPL  LV1
              ;
897E A0 3F             LDY  #63
8980 B9 00 DC LV2      LDA  $DC00,Y
8983 91 FD             STA  (USE),Y  ;NACH ZIELADRESSE SPEICHERN
8985 88                DEY
8986 10 F8             BPL  LV2
              ;
```

```
                ;ADRESSEN PLUS 64:
                ;
8988 A5 FD              LDA  USE
898A 18                 CLC
898B 69 40              ADC  #64
898D 85 FD              STA  USE
898F 90 05              BCC  LV3
8991 E6 FE              INC  USE+1
8993 E6 AF              INC  SHP+1
8995 18                 CLC
8996 69 08      LV3     ADC  #8
8998 85 AE              STA  SHP
899A 90 02              BCC  LV4
899C E6 AF              INC  SHP+1
                ;
899E CA         LV4     DEX            ;5TELABSCHN.ZAEHLER
899F D0 D3              BNE  LIVER     ;NAECHSTEN ABSCHNITT
                ;
                ;ZWISCHENGESPEICHERTEN GRAPHIKRAND
                ;HINUEBERROLLEN:
                ;
89A1 A5 FD              LDA  USE
89A3 38                 SEC
89A4 E9 08              SBC  #8
89A6 85 AC              STA  ADL
89A8 A5 FE              LDA  USE+1
89AA E9 00              SBC  #0
89AC 85 AD              STA  ADH       ;USE-8 -> ADL/ADH
                ;
89AE A0 07              LDY  #7
89B0 B6 57      LV5     LDX  D0,Y      ;BYTE HOLEN
89B2 8A                 TXA
89B3 91 AC              STA  (ADL),Y ;ERSTE 8 BYTES
89B5 88                 DEY
89B6 10 F8              BPL  LV5
                ;
89B8 18                 CLC            ;RICHTUNGSFLAG
89B9 20 43 8A           JSR  MCSC      ;BEI MC AUCH FARBRAM VERSCH.
                ;
                ;
                ;VIDEORAM LINKS VERSCHIEBEN:
                ;****************************
                ;
89BC A0 00      LVC     LDY  #0
89BE B1 61              LDA  ($61),Y
89C0 AA                 TAX            ;ERSTES BYTE MERKEN
89C1 24 96              BIT  FLG2
89C3 10 02              BPL  LV8       ;ROLLEN
89C5 A6 15              LDX  XK+1      ;NACHSCHUB
                ;
```

```
89C7 A0 27    LV8     LDY  #39       ;40 BYTES
89C9 B1 61    LV6     LDA  ($61),Y   ;BYTE HOLEN
89CB 48               PHA            ;MERKEN
89CC 8A               TXA            ;LETZTES BYTE WIEDERHOLEN
89CD 91 61            STA  ($61),Y   ;UND DORT ABSPEICHERN
89CF 68               PLA
89D0 AA               TAX
89D1 88               DEY
89D2 10 F5            BPL  LV6       ;NAECHSTES BYTE
              ;
89D4 18               CLC
89D5 A5 61            LDA  $61
89D7 69 28            ADC  #40
89D9 85 61            STA  $61
89DB 90 02            BCC  LV7
89DD E6 62            INC  $62       ;ADRESSE NAECHSTE ZEILE
89DF 60       LV7     RTS
              ;
              ;
              ;
              ;HGR/MC-ZEILE RECHTSVERSCHIEBEN:
              ;*********************************
              ;
              ;
              ;5TEL-ABSCHNITTSSCHLEIFE:
              ;
89E0 A5 FD    REVER   LDA  USE
89E2 38               SEC
89E3 E9 38            SBC  #56       ;USE ANS BLOCKENDE
89E5 85 AE            STA  SHP       ;SHP=USE+8
89E7 B0 03            BCS  RV1
89E9 C6 FE            DEC  USE+1
89EB 38               SEC
89EC A4 FE    RV1     LDY  USE+1
89EE 84 AF            STY  SHP+1
89F0 E9 08            SBC  #8
89F2 85 FD            STA  USE
89F4 B0 02            BCS  RV2
89F6 C6 FE            DEC  USE+1
              ;
89F8 A0 3F    RV2     LDY  #63       ;64 BYTES
89FA B1 FD    RV3     LDA  (USE),Y
89FC 99 00 DC         STA  $DC00,Y   ;ZWISCHENSPEICHERN
89FF 88               DEY
8A00 10 F8            BPL  RV3
              ;
8A02 A0 3F            LDY  #63
8A04 B9 00 DC RV4     LDA  $DC00,Y
8A07 91 AE            STA  (SHP),Y   ;UND UEBERTRAGEN
8A09 88               DEY
8A0A 10 F8            BPL  RV4
```

```
                ;
8A0C CA                  DEX          ;ZAEHLER
8A0D D0 D1               BNE  REVER   ;NAECHSTEN ABSCHNITT
                ;
                ;ROLLEN/SCROLLEN:
                ;
8A0F A0 07               LDY  #7
8A11 B6 57      RV5      LDX  D0,Y
8A13 8A                  TXA
8A14 91 AE               STA  (SHP),Y
8A16 88                  DEY
8A17 10 F8               BPL  RV5     ;BYTES AUS ZWISCHENSPEICHER
                ;
8A19 38                  SEC          ;RICHTUNGSFLAG
8A1A 20 43 8A            JSR  MCSC    ;BEI MC AUCH FARBRAM VERSCHIEBEN
                ;
                ;
                ;VIDEORAM RECHTSVERSCHIEBEN:
                ;****************************
                ;
8A1D 38         RVC      SEC
8A1E A5 61               LDA  $61
8A20 E9 28               SBC  #40
8A22 85 61               STA  $61
8A24 B0 02               BCS  RV7
8A26 C6 62               DEC  $62     ;ADRESSE EINE ZEILE WEITER
                ;
8A28 A0 28      RV7      LDY  #40
8A2A B1 61               LDA  ($61),Y ;ZU ROLLENDES BYTE
8A2C AA                  TAX
8A2D 24 96               BIT  FLG2
8A2F 10 02               BPL  RV8     ;ROLLEN
8A31 A6 15               LDX  XK+1    ;NACHSCHUB
8A33 A0 01      RV8      LDY  #1
                ;
8A35 B1 61      RV6      LDA  ($61),Y ;BYTE HOLEN
8A37 48                  PHA          ;MERKEN
8A38 8A                  TXA          ;LETZTES BYTE HOLEN
8A39 91 61               STA  ($61),Y ;AN DIESE STELLE ABLEGEN
8A3B 68                  PLA
8A3C AA                  TAX          ;GEMERKTES BYTE FUER NAECHSTE SCHLEIFE
VORBEREITEN
8A3D C8                  INY
8A3E C0 29               CPY  #41
8A40 D0 F3               BNE  RV6     ;NAECHSTES BYTE
                ;
8A42 60                  RTS          ;FERTIG
                ;
                ;
```

```
              ;FARBRAM VERSCHIEBEN:
              ;********************
              ;
8A43 24 02    MCSC      BIT  GFLAG
8A45 10 06              BPL  MCSC1   ;KEIN MC => NICHT VERSCHIEBEN
8A47 AD 1C C6           LDA  COL3    ;MC
8A4A 20 53 8A           JSR  FABRSC  ;FARBRAM VERSCHIEBEN
8A4D AD 55 C6 MCSC1     LDA  COLOR
8A50 85 15              STA  XK+1    ;NACHSCHUB F. VIDEORAM
8A52 60                 RTS
              ;
              ;
8A53 85 15    FABRSC    STA  XK+1    ;ROLLBYTE MERKEN
8A55 A5 62              LDA  $62
8A57 29 03              AND  #3
8A59 0D 51 C6           ORA  FABMEM  ;ADRESSE FARBRAM
8A5C 4C 6A 89           JMP  LGMV1   ;FARBRAM VERSCHIEBEN
              ;
              ;
              ;
              ;
              ;UEBERSICHT GRAPHIKFLAGS:
              ;************************
              ;
              ;GFLAG:
              ; BIT7=0: MULTICOLORMODUS AUS
              ;     =1: MULTICOLORMODUS EIN
              ; BIT6=0: AUSSTEIGERFLAG LGR/SPRITES AUS
              ;     =1: AUSSTEIGERFLAG EIN
              ; BIT5=0: MISCHANZEIGE AUS
              ;     =1: MISCHANZEIGE EIN (RASTER-IRQ)
              ; BIT4=0: %
              ;     =1:
              ; BIT3=0: HGR/MC-ANZEIGE AUSGESCHALTET
              ;     =1: HGR/MC-ANZEIGE EINGESCHALTET
              ; BIT2=0: %
              ;     =1:
              ; BIT1=0: %
              ;     =1:
              ; BIT0=0: LGR WIRD BEFEHLIGT
              ;     =1: MC/HGR WIRD BEFEHLIGT
              ;
              ;
              ;GFLAG2:
              ; BIT7=0: ;SEITE 2 ANGEZEIGT
              ;     =1: ;SEITE 1 ANGEZEIGT
              ; BIT3=0: ;SEITE 1 BEFEHLIGT
              ;     =1: ;SEITE 2 BEFEHLIGT
              ;
              ;
              ;
```

```
              ;
              ;
              ;TABELLE DER GRAPHIKFLAGKOMBINATIONEN
              ;GEORDNET NACH MODUSNUMMERN:
              ;************************************
              ;
8A5F 01       GMTAB    .BYT %00000001
8A60 08                .BYT %00001000
8A61 88                .BYT %10001000
8A62 09                .BYT %00001001
8A63 89                .BYT %10001001
8A64 00                .BYT %00000000
8A65 80                .BYT %10000000
8A66 29                .BYT %00101001
8A67 28                .BYT %00101000
              ;
              ;
              ;
              ;TABELLE DER GRAPHIKSEITENFLAGKOMBINATIONEN
              ;GEORDNET NACH MODUSNUMMERN:
              ;******************************************
              ;
8A68 88       GMTAB2   .BYT %10001000
8A69 80                .BYT %10000000
8A6A 00                .BYT %00000000
8A6B 08                .BYT %00001000
              ;
              ;
              ;
              ;GRAPHIKMODUS INITIALISIERUNG:
              ;*****************************
              ;
              ;
8A6C A5 02    GMODEI   LDA  GFLAG    ;GRAPHIKFLAG LADEN
8A6E 29 20             AND  #$20
8A70 F0 03             BEQ  GM000    ;KEIN RASTER-IRQ
8A72 20 AE 93          JSR  SOFF16   ;SATZ 2 SPRITES AUS
8A75 A5 02    GM000    LDA  GFLAG    ;(FALLS GERADE TEXT)
8A77 10 0D             BPL  GM00     ;KEIN MC
              ;
              ;MC EIN:
8A79 29 08             AND  #8
8A7B F0 09             BEQ  GM00     ;NICHT ANGEZEIGT
              ;
8A7D AD 1D C6          LDA  TCOL
8A80 8D 21 D0          STA  V+33     ;FARBE 3
8A83 20 43 8C          JSR  ROTCOL   ;MC => FARBRAM HOLEN
              ;
8A86 A9 FF    GM00     LDA  #$FF
8A88 85 FD             STA  USE      ;KEINE SEITENANGABE-FLAG
8A8A 60       RTS.     RTS
```

```
              ;
              ;
              ;
              ;**********************
              ;**                  **
              ;**  BEFEHL: GMODE   **
              ;**                  **
              ;**********************
              ;
              ;
8A8B 20 6C 8A GMODE    JSR  GMODEI  ;INITIALISIERUNG
8A8E 20 79 00          JSR  CHRGOT
8A91 C9 2C             CMP  #","    ;KOMMA?
8A93 F0 0D             BEQ  GM01    ;JA
8A95 20 9E B7          JSR  GETBYT  ;HOLE SEITENNUMMER
8A98 8A                TXA
8A99 29 03             AND  #$03    ;KORREKTUR
8A9B 85 FD             STA  USE     ;SEITE
8A9D 20 79 00          JSR  CHRGOT
8AA0 F0 0E             BEQ  GM2     ;KEIN ZEICHEN MEHR
              ;
              ;GRAPHIKMODUS:
              ;
8AA2 20 F1 B7 GM01     JSR  CHKGET  ;HOLE GRAPHIKMODUS
8AA5 E0 09             CPX  #9
8AA7 90 02             BCC  GM1     ;KORREKTUR
8AA9 A2 00             LDX  #0      ;BEI FEHLER: LGR
8AAB BD 5F 8A GM1      LDA  GMTAB,X ;HOLE ENTSPRECHENDE FLAGKOMBINATION AUS
TABELLE
8AAE 85 02             STA  GFLAG   ;GRMODUS, NEUES GRAPHIKFLAG
              ;
              ;SEITENBEARBEITUNG:
              ;
8AB0 A4 FD    GM2      LDY  USE        ;SEITENNUMMER
8AB2 30 19             BMI  GM2C       ;KEINE SEITENANGABE
8AB4 B9 68 8A          LDA  GMTAB2,Y   ;ENTSPRECHENDE FLAGKOMBINATION AUS
TABELLE
8AB7 48                PHA          ;MERKEN
8AB8 4D 1E C6          EOR  GFLAG2  ;VERAENDERTE BITS=1
8ABB 48                PHA
8ABC 10 03             BPL  GM2A    ;->SEITENANZEIGE UNVERAENDERT
8ABE 20 5A 7D          JSR  TRANS   ;SEITENAUSTAUSCH, DA VERAENDERT
8AC1 68       GM2A     PLA
8AC2 29 08             AND  #8
8AC4 F0 03             BEQ  GM2B    ;->KEINE AENDERUNG IN DER BEFEHLIGUNG
8AC6 20 9C 7D          JSR  TRSBEF  ;BEFEHLS-ZEIGERAUSTAUSCH, DA VERAENDERUNG
8AC9 68       GM2B     PLA          ;NEUE FLAGKOMBINATION HOLEN
8ACA 8D 1E C6          STA  GFLAG2  ;NEUES FLAG
              ;
```

```
              ;MISCHANZEIGE:
              ;
8ACD A5 02    GM2C     LDA  GFLAG
8ACF 29 20             AND  #$20
8AD1 F0 1B             BEQ  GM2D.   ;KEIN RASTER-IRQ
8AD3 20 FD AE          JSR  CHKCOM  ;AUF KOMMA TESTEN
8AD6 20 64 8C          JSR  GETRAS  ;HOLE STREIFENBREITENPARAMETER
8AD9 2C 94 C6          BIT  RASFLG  ;RASTER-IRQ-FLAG
8ADC 10 02             BPL  GM2C.   ;Z.ZT.KEIN RASTER-IRQ AN
8ADE 70 AA             BVS  RTS.    ;Z.ZT.BEREITS TEXT-IRQ AN
              ;
              ;TEXT/HGR-IRQ EINSCHALTEN:
              ;
8AE0 78       GM2C.    SEI
8AE1 A9 C0             LDA  #$C0
8AE3 8D 94 C6          STA  RASFLG  ;TEXT-IRQ EIN (DAMIT EVT. SPRITE-IRQ AUS)
8AE6 A9 81             LDA  #$81
8AE8 8D 1A D0          STA  V+26    ;INTERRUPT MASK REGISTER STELLEN
8AEB 58                CLI
8AEC D0 0C             BNE  GRAPHM  ;UNBED.
              ;
              ;
8AEE 8D 94 C6 GM2D.    STA  RASFLG  ;RASTER-IRQ-FLAG LOESCHEN
8AF1 8D 1A D0          STA  V+26    ;INTERRUPT MASK REGISTER (IMR)
8AF4 A5 02    GM2D     LDA  GFLAG
8AF6 29 08             AND  #$08
8AF8 F0 55             BEQ  TEXTM   ;LGR BZW. TEXT
              ;
              ;HGR EINSCHALTEN:
              ;
8AFA 20 BC 8B GRAPHM   JSR  PCOLHG  ;COL1=HIGH-/BCOL=LOW-BITS IN VIDEORAM
8AFD A9 3B    GRAPM1   LDA  #$3B    ;BITMAP-MODE
8AFF A2 08             LDX  #$08    ;VIDEORAM:$C000/GRAPHIK:$E000
8B01 A0 94             LDY  #$94    ;OBERSTE BITS=%11
8B03 8C 00 DD          STY  W       ;$DD00
8B06 8E 18 D0          STX  V+24    ;GRAPHIKADRESSE FESTLEGEN
8B09 8D 11 D0          STA  V+17    ;STEUERREG.1
8B0C AD 88 C6          LDA  SPRIT1
8B0F 8D 15 D0          STA  V+21    ;SPRITE ENABLE
8B12 A9 08             LDA  #$08    ;KEIN MC-MODE
8B14 8D 16 D0          STA  V+22    ;STEUERREGISTER 2
8B17 AD 21 D0          LDA  V+33    ;TEXTFARBE
8B1A 8D 1D C6          STA  TCOL    ;TEXTCOLOR
8B1D 24 02             BIT  GFLAG
8B1F 30 01             BMI  GM3     ;MC-MODE
8B21 60                RTS
              ;
```

```
              ;MULTICOLORMODUS EINSCHALTEN:
              ;
8B22 20 43 8C GM3       JSR  ROTCOL   ;FARBRAM RETTEN
8B25 AD 16 D0           LDA  V+22     ;MULTICOLOR
8B28 09 10              ORA  #16
8B2A 8D 16 D0           STA  V+22
8B2D 20 CA 8B           JSR  PCOLMC   ;COL2=L-/COL1=H-BITS IN VIDEORAM
8B30 AD 19 C6           LDA  BCOL     ;HGRUNDFARBE
8B33 8D 21 D0           STA  V+33     ;IN MC HGRUNDFARBE
              ;
              ;FARBRAM MIT FARBE FUELLEN:
              ;
8B36 AD 1C C6 FRFILL    LDA  COL3     ;COLOR3 AUS FARBRAM
8B39 48                 PHA           ;MERKEN
8B3A A0 00              LDY  #0
8B3C A9 D8              LDA  #$D8
8B3E 20 20 7D           JSR  STU      ;$D800 NACH USE
8B41 A2 04              LDX  #4
8B43 68                 PLA           ;FARBE
8B44 91 FD    GM5       STA  (USE),Y ;NACH FARBRAM
8B46 C8                 INY
8B47 D0 FB              BNE  GM5
8B49 E6 FE              INC  USE+1
8B4B CA                 DEX
8B4C D0 F6              BNE  GM5
8B4E 60                 RTS
              ;
              ;TEXTMODUS EINSCHALTEN:
              ;
8B4F A9 1B    TEXTM     LDA  #$1B     ;LOESCHE BIT-MODE
8B51 A2 14              LDX  #$14     ;BASISADRESSE=$0400
8B53 A0 97              LDY  #$97     ;OBERSTE BITS=%00
8B55 8C 00 DD           STY  W        ;$DD00
8B58 8E 18 D0           STX  V+24
8B5B 8D 11 D0           STA  V+17     ;STEUERREG.1
8B5E A9 00              LDA  #0
8B60 8D 15 D0           STA  V+21     ;SPRITE ENABLE
8B63 A9 08              LDA  #8
8B65 8D 16 D0           STA  V+22     ;STEUERREG.2-MC-MODE AUS
8B68 60                 RTS
              ;
              ;
              ;
```

```
              ;*********************
              ;**                 **
              ;**  BEFEHL: SCOL=  **
              ;**                 **
              ;*********************
              ;
              ;
8B69 20 9E B7 SCOLOR   JSR  GETBYT
8B6C E0 04             CPX  #4      ;FARBNR.
8B6E 90 02             BCC  SCOL3
8B70 A2 01             LDX  #1      ;KORREKTUR
8B72 CA       SCOL3    DEX
8B73 30 73             BMI  BCOLO.  ;NR.=0 => HINTERGRUND
8B75 20 FD AE          JSR  CHKCOM  ;AUF KOMMA TESTEN
8B78 E8                INX          ;FARBE WIEDER INCREMIEREN
8B79 20 A5 8B          JSR  TTTT    ;FARBEN HOLEN UND EINSTELLEN
              ;
8B7C A4 AB             LDY  MSK     ;FARBNUMMER HOLEN
8B7E B9 5C 8C          LDA  MSKHG,Y ;VIDEORAMAUFBAU-MASKE
8B81 85 AB             STA  MSK     ;MERKEN
8B83 20 BC 8B          JSR  PCOLHG  ;COLOR-SPEICHER FESTLEGEN
8B86 24 02             BIT  GFLAG
8B88 10 0C             BPL  JMPSCL  ;HGR
              ;
8B8A B9 60 8C          LDA  MSKMC,Y ;MC
8B8D 85 AB             STA  MSK
8B8F 20 CA 8B          JSR  PCOLMC  ;WIE OBEN
8B92 C0 03             CPY  #3      ;FARBE 3?
8B94 F0 A0             BEQ  FRFILL  ;FARBRAM ANFUELLEN
              ;
8B96 4C 0D 8C JMPSCL   JMP  SCOL1   ;BILDSCHIRM FAERBEN
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: PCOL=  **
              ;**                 **
              ;*********************
              ;
              ;
8B99 20 9E B7 PCOLOR   JSR  GETBYT  ;HOLE PARAMETER
8B9C E0 05             CPX  #5
8B9E 90 02             BCC  PCL1
8BA0 A2 01             LDX  #1      ;KORREKTUR
8BA2 20 FD AE PCL1     JSR  CHKCOM  ;TESTE AUF KOMMA
              ;
```

```
8BA5 8A       TTTT     TXA            ;FARB-NR.
8BA6 48                PHA            ;MERKEN
8BA7 20 9E B7          JSR  GETBYT    ;FARBE NACH X HOLEN
8BAA 68                PLA            ;HOLE FARB-NR.
8BAB C9 04             CMP  #4
8BAD F0 29             BEQ  SLCOLO    ;NR=4 (FUER MC)
8BAF A8                TAY            ;FARB-NR.
8BB0 8A                TXA            ;FARBE
8BB1 29 0F             AND  #$0F      ;KORREKTUR
8BB3 99 19 C6          STA  BCOL,Y    ;ENTSPRECHENDE FARBE SPEICHERN
8BB6 84 AB             STY  MSK       ;POINTER AUF TABELLE
              ;
8BB8 24 02             BIT  GFLAG
8BBA 30 0E             BMI  PCOLMC    ;MC
              ;
              ;
              ;COLOR FUER HGR FESTLEGEN:
              ;*************************
              ;
8BBC AD 1A C6 PCOLHG   LDA  COL1
8BBF 0A                ASL  A
8BC0 0A                ASL  A
8BC1 0A                ASL  A
8BC2 0A                ASL  A         ;COL1 NACH OBERSTEM NIBBLE
8BC3 0D 19 C6          ORA  BCOL      ;BCOL NACH UNTERSTEM NIBBLE
8BC6 8D 55 C6          STA  COLOR
8BC9 60                RTS
              ;
              ;
              ;COLOR FUER MC FESTLEGEN:
              ;************************
              ;
8BCA AD 1A C6 PCOLMC   LDA  COL1
8BCD 0A                ASL  A
8BCE 0A                ASL  A
8BCF 0A                ASL  A
8BD0 0A                ASL  A         ;COL1 NACH OBERSTEM NIBBLE
8BD1 0D 1B C6          ORA  COL2      ;COL2 NACH UNTERSTEM NIBBLE
8BD4 8D 55 C6          STA  COLOR
8BD7 60                RTS
              ;
              ;
              ;MC-FARBNUMMER SETZEN:
              ;*********************
              ;
8BD8 8A       SLCOLO   TXA
8BD9 29 03             AND  #$03
8BDB 8D 1F C6          STA  LCOLO     ;MC-FARBE
8BDE 60                RTS
              ;
              ;
```

```
                ;
                ;***********************
                ;**                   **
                ;**  BEFEHL: COLOR=   **
                ;**                   **
                ;***********************
                ;
                ;
8BDF 20 9E B7 BCOLOR   JSR  GETBYT  ;RAHMENFARBE NACH X
8BE2 20 F0 95          JSR  GET     ;15ER ZAHL (S.TOENE)
8BE5 8D 20 D0          STA  V+32    ;RAHMENFARBE
                ;
8BE8 20 F1 B7 BCOLO.   JSR  CHKGET  ;HINTERGRUNDFARBE NACH X
8BEB 8A                TXA
8BEC 29 0F             AND  #$0F    ;KORREKTUR
8BEE AA                TAX
                ;
8BEF A5 02             LDA  GFLAG
8BF1 4A                LSR  A       ;BIT 0 NACH CARRY
8BF2 90 07             BCC  BC1     ;GRAPHIK BEFEHLIGT
                ;
                ;TEXTHINTERGRUNDFARBE:
                ;
8BF4 8E 1D C6          STX  TCOL    ;NACH TEXTHINTERGRUNDFARBE
8BF7 8E 21 D0          STX  V+33
8BFA 60                RTS
                ;
                ;GRAPHIK-HINTERGRUNDFARBE SETZEN:
                ;
8BFB 8E 19 C6 BC1      STX  BCOL
8BFE A5 02             LDA  GFLAG
8C00 30 36             BMI  GM12    ;MC
                ;
8C02 A9 F0             LDA  #$F0
8C04 85 AB             STA  MSK     ;VIDEORAMAUFBAU
8C06 20 BC 8B          JSR  PCOLHG  ;COLOR FUER HGR
                ;
8C09 A5 02      SCOL   LDA  GFLAG
8C0B 30 2B             BMI  GM12    ;MC
                ;
8C0D A2 03      SCOL1  LDX  #3
8C0F 20 13 7D          JSR  STUCOL  ;FARBADRESSE NACH USE
8C12 84 97             STY  FLG
8C14 A5 AB             LDA  MSK
8C16 49 FF             EOR  #$FF
8C18 2D 55 C6          AND  COLOR
8C1B 85 96             STA  FLG2
                ;
```

```
8C1D B1 FD    GM9     LDA  (USE),Y
8C1F 25 AB            AND  MSK     ;FARBNIBBLE AUSBLENDEN
8C21 05 96            ORA  FLG2    ;FARBE
8C23 91 FD            STA  (USE),Y ;SETZEN
8C25 C8               INY
8C26 C4 97            CPY  FLG     ;BIS FLG
8C28 D0 F3            BNE  GM9
8C2A E6 FE            INC  USE+1
8C2C CA               DEX
8C2D F0 03            BEQ  GM9.
8C2F 10 EC            BPL  GM9
8C31 60               RTS
              ;
8C32 A2 E8    GM9.    LDX  #$E8
8C34 86 97            STX  FLG     ;SPRITEVEKTOREN SCHUETZEN
8C36 D0 E5            BNE  GM9     ;UNBEDINGT
              ;
              ;
              ;MULTICOLOR HINTERGRUNDFARBE:
              ;
8C38 29 08    GM12    AND  #8      ;A=GFLAG
8C3A F0 06            BEQ  GM13    ;MC NICHT ANGEZEIGT!
8C3C AD 19 C6         LDA  BCOL
8C3F 8D 21 D0         STA  V+33    ;FARBE SETZEN
8C42 60       GM13    RTS
              ;
              ;
              ;FARBRAM RETTEN BZW. HOLEN:
              ;***************************
              ;
8C43 AD 51 C6 ROTCOL  LDA  FABMEM
8C46 AE 54 C6         LDX  FABMM
8C49 8D 54 C6         STA  FABMM
8C4C 8E 51 C6         STX  FABMEM  ;FARBRAMADRESSEN WECHSELN
8C4F A9 00            LDA  #0
8C51 85 FD            STA  USE
8C53 85 AC            STA  ADL
8C55 A9 D8            LDA  #PA1F
8C57 A0 CC            LDY  #PA2F   ;ZIEL UND QUELLE FESTLEGEN
8C59 4C D5 7D         JMP  TRCOL   ;FARBRAM RETTEN/HOLEN
              ;
              ;
```

```
              ;TABELLEN FUER VIDEORAM-FARBMASKEN:
              ;**********************************
              ;
              ;HGR:
              ;
8C5C F0 0F FF MSKHG    .BYT $F0,$0F,$FF,$FF
              ;HINTERG/COL1/%/ %
              ;
              ;MC:
              ;
8C60 FF 0F F0 MSKMC    .BYT $FF,$0F,$F0,$FF
              ;    % / % /COL2/COL3
              ;
              ;
              ;GMODE-PARAMETER FUER MISCHANZEIGE HOLEN:
              ;***************************************
              ;
8C64 20 76 8C GETRAS   JSR  GETRA1  ;1. PARAMETER
8C67 48                PHA
8C68 20 FD AE          JSR  CHKCOM
8C6B 20 76 8C          JSR  GETRA1  ;2. PARAMETER
8C6E 8D 38 C6          STA  RAST2
8C71 68                PLA
8C72 8D 37 C6          STA  RAST1
8C75 60                RTS
              ;
8C76 20 9E B7 GETRA1   JSR  GETBYT  ;RASTERZEILE
8C79 8A                TXA
8C7A 18                CLC
8C7B 69 28             ADC  #40     ;+40 IN RASTERZEILE UMRECHNEN
8C7D 60                RTS
              ;
              ;
              ;
              ;
              ;
              ;*******************
              ;**               **
              ;**  BEFEHL: DRAW **
              ;**               **
              ;*******************
              ;
              ;
8C7E 20 67 7C DRAW     JSR  STFLG   ;ZEICHENFLAGS HOLEN
              ;
```

```
               ;DEFINITIONSSTRING HOLEN:
               ;
8C81 20 9E AD            JSR  FRMEVL  ;NAECHSTEN AUSDRUCK HOLEN
8C84 20 8F AD            JSR  CHKSTR  ;TESTE AUF STRING
8C87 20 82 B7            JSR  $B782   ;HOLE STRINGPARAMETER
8C8A A5 22               LDA  $22
8C8C A6 23               LDX  $23     ;ADRESSE STRING
8C8E 48                  PHA
8C8F 8A                  TXA
8C90 48                  PHA          ;RETTEN
8C91 98                  TYA          ;LAENGE STRING
8C92 48                  PHA          ;RETTEN
8C93 08                  PHP          ;FLAGS RETTEN
               ;
8C94 A5 02               LDA  GFLAG
8C96 4A                  LSR  A
8C97 90 05               BCC  DR1     ;KEIN TEXT
               ;
               ;TEXT-MODUS:
               ;
8C99 2A                  ROL  A
8C9A 09 40               ORA  #$40
8C9C 85 02               STA  GFLAG   ;AUSSTEIGERMODE
               ;
               ;KOORDINATEN HOLEN:
               ;
8C9E 20 79 00 DR1        JSR  CHRGOT
8CA1 C9 91               CMP  #AT     ;AT-TOKEN?
8CA3 D0 1C               BNE  D9      ;NEIN->KEINE KOORDINATEN
8CA5 20 73 00            JSR  CHRGET
8CA8 20 FF 7F            JSR  TESCOR  ;HOLE KOORDINATEN NACH A,X,Y
8CAB 28                  PLP
8CAC 08                  PHP          ;WAR STRINGLAENGE=0?
8CAD F0 18               BEQ  DR2     ;JA
8CAF 48                  PHA          ;KOORDINATE IN A RETTEN
8CB0 A5 02               LDA  GFLAG
8CB2 4A                  LSR  A
8CB3 68                  PLA
8CB4 90 08               BCC  D8      ;GRAPHIKMODUS
               ;TEXTMODUS:
8CB6 8D 3E C6            STA  X1L
8CB9 8C 3C C6            STY  Y1
8CBC B0 09               BCS  DR2     ;UNBEDINGT
               ;
8CBE 20 AA 81 D8         JSR  HPOSN   ;STARTPUNKTADRESSE BERECHNEN
8CC1 20 63 88 D9         JSR  GETE5   ;E5 ERRECHNEN
8CC4 8D 15 C6            STA  E5.ZW   ;ZWISCHENSPEICHERN
               ;
```

```
                ;DRAW AUSFUEHREN:
                ;
8CC7 28         DR2     PLP             ;FLAGS VOM STACK
8CC8 68                 PLA             ;STRINGLAENGE
8CC9 F0 1A              BEQ  DR3        ;=0 => ENDE!
8CCB 85 14              STA  XK         ;LAENGE MERKEN
8CCD 68                 PLA
8CCE 85 AF              STA  SHP+1
8CD0 68                 PLA
8CD1 85 AE              STA  SHP        ;STRINGADRESSE
8CD3 AD 14 C6           LDA  RT         ;ROTATIONSWINKEL (ROT)
8CD6 20 EE 8C           JSR  DRAW.1     ;ZEICHNE FIGUR
                ;
                ;TERMINIEREN:
                ;
8CD9 A5 02              LDA  GFLAG
8CDB 29 BD              AND  #%10111101
8CDD 85 02              STA  GFLAG      ;AUSSTEIGERFLAG AUS
8CDF 20 F7 7C           JSR  TZA        ;ADRESSE NACH ZWISCHENSPEICHER
8CE2 4C C4 7F           JMP  PLTVAR     ;TEST?
                ;
                ;BEENDEN BEI STRINGLAENGE=0:
                ;
8CE5 A5 02      DR3     LDA  GFLAG
8CE7 29 BD              AND  #%10111101
8CE9 85 02              STA  GFLAG
8CEB 68                 PLA
8CEC 68                 PLA             ;STRINGADRESSE VOM STACK
8CED 60                 RTS             ;ENDE
                ;
                ;
                ;
                ;DRAW-FIGUR ZEICHNEN:
                ;*********************
                ;
                ;ROTATION BEARBEITEN:
                ;
8CEE AA         DRAW.1  TAX             ;ROTATIONSWINKEL->X
8CEF 4A                 LSR  A          ;DEN DURCH 90 GRAD TEILBAREN
8CF0 4A                 LSR  A          ;ANTEIL DES WINKELS ALS EINFACHE
8CF1 4A                 LSR  A          ;RICHTUNGSAENDERUNG MERKEN
8CF2 4A                 LSR  A          ;/16
8CF3 85 5A              STA  D3         ;UNTERES NIBBLE
8CF5 8A                 TXA             ;ROT
8CF6 29 0F              AND  #$0F       ;OBERES NIBBLE
8CF8 AA                 TAX             ;ALS OFFSET
8CF9 BC 68 8D           LDY  RDAT,X     ;HOLE SINUS AUS TABELLE
8CFC 84 57              STY  D0         ;NACH D0
8CFE 49 0F              EOR  #$0F       ;WERTE UMKEHREN
8D00 AA                 TAX             ;ALS OFFSET
8D01 BC 69 8D           LDY  RDAT+1,X   ;COSINUS AUS TABELLE
```

```
8D04 C8                 INY
8D05 84 59              STY  D2        ;NACH D2
              ;
              ;DEFINITION ABFRAGEN UND ZEICHNEN:
              ;
8D07 AC 15 C6           LDY  E5.ZW   ;E5 (X-KOORD/8) -> Y
8D0A A2 00              LDX  #0
8D0C 8A                 TXA
8D0D 01 AE    D.0       ORA  (SHP,X) ;ZEICHEN AUS DEFINITIONSSTRING (EVT. MIT
PLOTFLG OR-EN)
8D0F C9 2E              CMP  #"."    ;PUNKT FUER PLOT?
8D11 D0 04              BNE  D.2     ;NEIN!
8D13 A9 04              LDA  #4      ;JA->PLOTFLG SETZEN
8D15 D0 29              BNE  D.1     ;UNBEDINGT
              ;
8D17 29 07    D.2       AND  #$07    ;SYNTAX BERICHTIGEN
8D19 85 58              STA  D1      ;HOCH/RUNTER/RECHTS/LINKS-FLAG
8D1B A2 80              LDX  #$80
8D1D 86 5B              STX  D4
8D1F 86 5C              STX  D5      ;STARTWERTE FUER D4/D5
8D21 AE 13 C6           LDX  E7      ;VERGROESSERUNGSFAKTOR ALS ZAEHLER NACH X
              ;
              ;VERGROESSERUNGSSCHLEIFE:
              ;
8D24 A5 5B    D.3       LDA  D4
8D26 38                 SEC
8D27 65 57              ADC  D0      ;ANGEPASSTEN SINUSWERT
8D29 85 5B              STA  D4      ;ZUM RECHTSGEDREHTEN
8D2B 90 04              BCC  D.4     ;RICHTUNGSANTEIL ADDIEREN
8D2D 20 4B 8D           JSR  DRAW2   ;CURSOR BEWEGEN
8D30 18                 CLC
8D31 A5 5C    D.4       LDA  D5
8D33 65 59              ADC  D2      ;ANGEPASSTEN COSINUSWERT
8D35 85 5C              STA  D5      ;ZUM LINKSGEDREHTEN
8D37 90 03              BCC  D.5     ;RICHTUNGSANTEIL ADDIEREN
8D39 20 4C 8D           JSR  DRAW3   ;CURSOR BEWEGEN MIT CARRY=1
8D3C CA       D.5       DEX          ;ZAEHLER-1
8D3D D0 E5              BNE  D.3
              ;
8D3F 8A                 TXA
8D40 E6 AE    D.1       INC  SHP
8D42 D0 02              BNE  D.6
8D44 E6 AF              INC  SHP+1   ;ZEIGER AUF STRING INCREMIEREN
8D46 C6 14    D.6       DEC  XK      ;ZEICHENZAEHLER
8D48 D0 C3              BNE  D.0
8D4A 60                 RTS
              ;
              ;
```

```
              ;CURSOR BEWEGEN UND PLOTTEN:
              ;****************************
              ;
8D4B 18       DRAW2    CLC
8D4C 84 93    DRAW3    STY  E5      ;Y-REGISTER IN E5 MERKEN
8D4E A5 58             LDA  D1      ;RICHTUNGSFLAG
8D50 29 04             AND  #4      ;PLOTFLG GESETZT?
8D52 F0 03             BEQ  DA      ;NEIN
8D54 20 18 82          JSR  PLT     ;PLOT-ZENTRALROUTINE
8D57 A5 58    DA       LDA  D1      ;RICHTUNGSFLAG
8D59 65 5A             ADC  D3      ;PLUS ROT-ANTEIL
8D5B 29 03             AND  #3
8D5D C9 02             CMP  #2      ;RICHTUNG ERMITTELN
8D5F 6A                ROR  A       ;FUER DIE VEKTORROUTINEN
8D60 B0 03             BCS  DRAW4
8D62 4C 33 87          JMP  U.O     ;UNTEN/OBEN
8D65 4C B2 87 DRAW4    JMP  R.L     ;LINKS/RECHTS
              ;
              ;
              ;ANGEPASSTE SINUS/COSINUS-TABELLE
              ;FUER ROTATIONSWINKEL:
              ;**********************************
              ;
8D68 FF FE FA RDAT     .BYT $FF,$FE,$FA,$F4,$EC,$E1,$D4,$C5,$B4
8D71 A1 8D 78          .BYT $A1,$8D,$78,$61,$49,$31,$18,$FF
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: SCALE  **
              ;**                 **
              ;*********************
              ;
              ;
8D79 20 9E B7 SCALE    JSR  GETBYT  ;HOLE 1. PARAMETER->X
8D7C 8E 14 C6          STX  RT      ;ROTATIONSWINKEL (ROT)
8D7F 20 F1 B7          JSR  CHKGET  ;2. PARAMETER->X
8D82 E8                INX
8D83 8E 13 C6          STX  E7      ;VERGROESSERUNGSFAKTOR (SCALE)
8D86 60                RTS
              ;
              ;
              ;
              ;
              ;
```

```
              ;TABELLE FUER STARTADRESSEN
              ;UND LAENGEN VON:
              ;TEXTRAM/GRAPHIKSP/VIDEOR/FARBRAM
              ;*********************************
              ;
              ;SEITE 1:
              ;
              ;TEXTRAM:
8D87 04       STATAB   .BYT $04     ;STARTADRESSE (HIGH-BYTE)
8D88 E8 03             .WOR 1000    ;LAENGE
              ;
              ;GRAPHIKSPEICHER 1:
8D8A E0                .BYT $E0
8D8B 40 1F             .WOR 8000
              ;
              ;PA1C:
8D8D C0                .BYT $C0
8D8E E8 03             .WOR 1000
              ;
              ;PA1F:
8D90 D8                .BYT $D8
8D91 E8 03             .WOR 1000
              ;
              ;
              ;SEITE 2:
              ;
              ;TEXTRAM:
8D93 04       STATB2   .BYT $04
8D94 E8 03             .WOR 1000
              ;
              ;GRAPHIKSPEICHER 2:
8D96 A0                .BYT $A0
8D97 40 1F             .WOR 8000
              ;
              ;PA2C:
8D99 C8                .BYT $C8
8D9A E8 03             .WOR 1000
              ;
              ;PA2F:
8D9C CC                .BYT $CC
8D9D E8 03             .WOR 1000
              ;
              ;
              ;
              ;TABELLE DER SECUNDAERBEFEHLE:
              ;******************************
              ;
8D9F 4C 47 43 GETTAB   .ASC "LGCF"
              ;
              ;
              ;
```

```
              ;
              ;
              ;GSAVE/GLOAD-MODUS UND -SEITE HOLEN:
              ;************************************
              ;
8DA3 A2 00    DISK     LDX  #0
8DA5 86 AE             STX  SHP      ;MODIFLAG
8DA7 86 08             STX  HKFLG2   ;FIRST-FLAG LOESCHEN
8DA9 A0 00             LDY  #0       ;BLOCKZAEHLER
8DAB C9 2C    DI4      CMP  #","     ;KOMMA?
8DAD F0 1F             BEQ  DI1      ;JA->FERTIG
              ;
              ;LGCF IN TABELLE SUCHEN:
              ;
8DAF A2 03             LDX  #3         ;ZAEHLER FUER TABELLE
8DB1 DD 9F 8D DI3      CMP  GETTAB,X   ;FINDE MODUS
8DB4 F0 06             BEQ  DI2        ;GEFUNDEN
8DB6 CA                DEX
8DB7 10 F8             BPL  DI3
              ;
8DB9 4C 08 AF DI5      JMP  SERR     ;SYNTAX ERROR
              ;
              ;NUMMER ALS CODE IN SHP SPEICHERN:
              ;
8DBC 8A       DI2      TXA           ;NUMMER VON 0-3 (2 BITS)
8DBD 4A                LSR  A        ;GANZ NACH LINKS SCHIEBEN
8DBE 6A                ROR  A        ;MIT ERSTEM BIT IN CARRY
8DBF 26 AE             ROL  SHP      ;NACH SHP ROLLEN
8DC1 0A                ASL  A        ;2. BIT NACH CARRY
8DC2 26 AE             ROL  SHP      ;IN MODIFLAG (SHP) ROLLEN
              ;
8DC4 20 73 00          JSR  CHRGET
8DC7 C8                INY           ;NAECHSTES ZEICHEN
8DC8 C0 05             CPY  #5
8DCA B0 ED             BCS  DI5      ;>= 5! (NICHT MEHR ALS 4 ZEICHEN)
8DCC 90 DD             BCC  DI4      ;NAECHSTEN (UNBED.)
              ;
              ;MODIFLAG LINKSBUENDIG MACHEN:
              ;
8DCE 8C 48 C6 DI1      STY  ZWIS     ;ZAHL DER BLOECKE MERKEN
8DD1 88                DEY           ;-1
8DD2 20 73 00          JSR  CHRGET   ;NAECHSTES ZEICHEN (HINTER KOMMA)
8DD5 98                TYA           ;Y->A (WERTE: VON 0-3)
8DD6 49 03             EOR  #3       ;WERTE VON 3-0
8DD8 A8                TAY           ;A->Y
8DD9 F0 09             BEQ  DI7      ;BEREITS LINKSBUENDIG
              ;
```

```
8DDB A5 AE              LDA  SHP      ;MODIFLAG
8DDD 0A        DI8      ASL  A        ;RUECKE BIS SCHLUSS
8DDE 0A                 ASL  A
8DDF 88                 DEY
8DE0 D0 FB              BNE  DI8
8DE2 85 AE              STA  SHP
               ;
               ;SEITE BEARBEITEN:
               ;
8DE4 20 9E B7  DI7      JSR  GETBYT   ;SEITE -> X
8DE7 CA                 DEX           ;AUS 1-2 MACH 0-1
8DE8 8A                 TXA
8DE9 4A                 LSR  A        ;SEITENNUMMER NACH CARRY
8DEA F0 03              BEQ  DI9      ;OK.
8DEC 4C 48 B2           JMP  QERR     ;SEITENNUMMER WAR GROESSER 1->ILLEGAL
QUANTITY
               ;
8DEF 6A        DI9      ROR  A        ;SEITENNUMMER IN 7. BIT
8DF0 4D 1E C6           EOR  GFLAG2   ;SEITE
8DF3 49 80              EOR  #$80
8DF5 0A                 ASL  A        ;CARRY=1: SEITE2//CARRY=0: SEITE1
8DF6 08                 PHP           ;ALS SEITENFLAG
               ;
               ;
8DF7 20 FD AE           JSR  CHKCOM
8DFA 20 D4 E1           JSR  GETPAR   ;FILEPARAMETER HOLEN
8DFD 28                 PLP           ;SEITENFLAG
8DFE 60        DI0      RTS
               ;
               ;
               ;STARTADRESSE UND LAENGE DES ZU
               ;UEBERTRAGENDEN BLOCKES HOLEN:
               ;********************************
               ;
8DFF 08        DISK2    PHP           ;SEITENFLAG
8E00 38                 SEC           ;FERTIG-FLAG
8E01 CE 48 C6           DEC  ZWIS     ;BLOCKZAEHLER
8E04 30 20              BMI  DI2.1    ;ALLE BLOECKE UEBERTRAGEN->FERTIG (MIT
CARRY=1)
               ;
8E06 A9 00              LDA  #0
8E08 06 AE              ASL  SHP
8E0A 2A                 ROL  A
8E0B 06 AE              ASL  SHP
8E0D 2A                 ROL  A        ;2 BITS DES MODIFLAG IN AKU ROLLEN
8E0E 85 FD              STA  USE      ;IN USE ZWISCHENSPEICHERN
8E10 0A                 ASL  A        ;A*2
8E11 18                 CLC
8E12 65 FD              ADC  USE      ;A*2+A = A*3
               ;
```

```
8E14 28                PLP                     ;SEITENFLAG
8E15 90 02             BCC  DI2.2              ;SEITE1
8E17 69 0B             ADC  #STATB2-STATAB-1   ;SEITE 2
              ;
8E19 AA       DI2.2    TAX               ;ALS OFFSET FUER TABELLE
8E1A BD 87 8D          LDA  STATAB,X     ;STARTADRESSE-HIGH
8E1D 48                PHA               ;MERKEN
8E1E BC 88 8D          LDY  STATAB+1,X   ;LAENGE-LOW
8E21 BD 89 8D          LDA  STATAB+2,X   ;LAENGE-HIGH
8E24 AA                TAX
8E25 18                CLC           ;NICHT-FERTIG-FLAG
8E26 68       DI2.1    PLA
8E27 60                RTS
              ;
              ;
              ;
              ;**********************
              ;**                  **
              ;**  BEFEHL: GSAVE   **
              ;**                  **
              ;**********************
              ;
              ;
8E28 20 A3 8D GSAVE    JSR  DISK     ;NAMEN,MODI,SEITE
8E2B 08                PHP           ;SEITENFLAG
8E2C A0 01             LDY  #1
8E2E 84 B9             STY  SECADR   ;SECUNDAER-ADRESSE
8E30 20 F2 8E          JSR  SRCHFN   ;FILENUMMER BESTIMMEN
8E33 20 8F F6          JSR  SAVOUT   ;"SAVING" AUSGEBEN
8E36 20 C1 E1          JSR  OPEN     ;DATEI OEFFNEN
8E39 A6 B8             LDX  FILENR   ;FILENUMMER
8E3B 20 18 E1          JSR  CHKOUT   ;KANAL FREI FUER AUSGABEGERAET
              ;
8E3E 28       GS1      PLP
8E3F 08                PHP           ;SEITENFLAG
8E40 20 FF 8D          JSR  DISK2    ;PARAMETER FUER MODI
8E43 B0 23             BCS  GL2.     ;FERTIG
8E45 20 77 8E          JSR  GSAVE.   ;SPEICHERN
8E48 4C 3E 8E          JMP  GS1      ;NAECHSTEN BLOCK SPEICHERN
              ;
              ;
              ;
```

```
              ;*********************
              ;**                 **
              ;**  BEFEHL: GLOAD  **
              ;**                 **
              ;*********************
              ;
              ;
8E4B 20 A3 8D GLOAD     JSR  DISK    ;NAMEN,MODI,SEITE
8E4E 08                 PHP          ;SEITENFLAG
8E4F A9 00              LDA  #0
8E51 85 B9              STA  SECADR  ;SEKUNDAERADRESSE
8E53 20 F2 8E           JSR  SRCHFN  ;FILENUMMER BESTIMMEN
8E56 20 AF F5           JSR  SRCHOU  ;"SEARCHING" AUSGEBEN
8E59 20 C1 E1           JSR  OPEN    ;DATEI OEFFNEN
8E5C A6 B8              LDX  FILENR  ;FILENUMMER
8E5E 20 1E E1           JSR  CHKIN   ;EINGABEGERAET SETZEN
              ;
8E61 28       GL1       PLP
8E62 08                 PHP          ;SEITENFLAG
8E63 20 FF 8D           JSR  DISK2   ;PARAMETER FUER MODI HOLEN
8E66 90 09              BCC  GL3     ;NOCH NICHT FERTIG
              ;
              ;
              ;TERMINIERUNG:
              ;
8E68 68       GL2.      PLA          ;STACKPOINTER EINSTELLEN
8E69 20 CC FF GL2       JSR  CLRCH   ;KANAL SCHLIESSEN
8E6C A5 B8              LDA  FILENR  ;FILENUMMER
8E6E 4C CC E1           JMP  CLOSE   ;DATEI SCHLIESSEN
              ;
              ;
8E71 20 B9 8E GL3       JSR  GLOAD.  ;BLOCK LADEN
8E74 4C 61 8E           JMP  GL1     ;NAECHSTEN BLOCK LADEN
              ;
              ;
              ;
              ;BLOCK AUF DISKETTE SPEICHERN:
              ;*****************************
              ;
              ;EINGANG:
              ;         A: STARTADRESSE (HIGH-BYTE)
              ;         X: BLOCKLAENGE (HIGH-BYTE)
              ;         Y: BLOCKLAENGE (LOW-BYTE)
              ;
              ;
```

```
8E77 84 57    GSAVE.   STY  D0        ;LAENGE-LOW
8E79 86 58             STX  D1        ;LAENGE-HIGH-BYTE
8E7B 20 20 7D          JSR  STU       ;STARTADRESSE->USE//Y=0
8E7E A9 00             LDA  #0
8E80 85 AF             STA  SHP+1     ;ZAEHLER-LOW-BYTE
8E82 24 08             BIT  HKFLG2    ;FIRST-FLAG
8E84 30 0C             BMI  GS0       ;NUR VOR 1. BLOCK 2 BYTES
8E86 20 0C E1          JSR  BASOUT    ;STARTADRESSE (LOW-BYTE)
8E89 A9 FF             LDA  #$FF
8E8B 85 08             STA  HKFLG2    ;FLAG SETZEN
8E8D A5 FE             LDA  USE+1     ;STARTADRESSE (HIGH-BYTE)
8E8F 20 0C E1          JSR  BASOUT    ;AUSGEBEN
              ;
              ;BLOCK AUSGEBEN:
              ;
8E92 A5 01    GS0      LDA  1
8E94 48                PHA
8E95 29 FD             AND  #$FD      ;FARBRAM + RAM AUSWAEHLEN
8E97 78                SEI
8E98 85 01             STA  1         ;ROM AUS
              ;
8E9A B1 FD             LDA  (USE),Y   ;BYTE HOLEN
8E9C AA                TAX            ;NACH X
8E9D 68                PLA
8E9E 85 01             STA  1         ;ROM AN
8EA0 58                CLI
8EA1 8A                TXA
8EA2 20 0C E1          JSR  BASOUT    ;BYTE UEBER ROM-ROUTINE AUSGEBEN
8EA5 C8                INY
8EA6 C4 AF             CPY  SHP+1     ;LOW-BYTE
8EA8 D0 E8             BNE  GS0
8EAA E6 FE             INC  USE+1     ;HIGH-BYTE
8EAC C6 58             DEC  D1
8EAE F0 03             BEQ  GS0.      ;LETZTER DURCHGANG
8EB0 10 E0             BPL  GS0
8EB2 60                RTS
              ;
8EB3 A6 57    GS0.     LDX  D0        ;LAENGE - LOW (<>0)
8EB5 86 AF             STX  SHP+1     ;BEWAHRE SPRITEVEKTOREN
8EB7 D0 D9             BNE  GS0       ;UNBEDINGT
              ;
              ;
              ;
```

```
               ;BLOCK VON DISKETTE LADEN:
               ;*************************
               ;
               ;EINGANG:
               ;        A: STARTADRESSE (HIGH-BYTE)
               ;        X: BLOCKLAENGE (HIGH-BYTE)
               ;        Y: BLOCKLAENGE (LOW-BYTE)
               ;
               ;
8EB9 84 57     GLOAD.   STY  D0       ;LAENGE-LOW
8EBB 86 58              STX  D1       ;LAENGE-HIGH
8EBD 20 20 7D           JSR  STU      ;STARTADRESSE->USE//Y=0
8EC0 A9 00              LDA  #0
8EC2 85 AF              STA  SHP+1    ;ZAEHLER (LOW-BYTE)
8EC4 85 93              STA  L.VFLG
8EC6 24 08              BIT  HKFLG2
8EC8 30 0D              BMI  GL00     ;NUR VORNE 2 BYTES
8ECA A9 FF              LDA  #$FF
8ECC 85 08              STA  HKFLG2   ;FLAG SETZEN
8ECE 20 12 E1           JSR  BASIN    ;STARTADRESSE
8ED1 20 12 E1           JSR  BASIN    ;WIRD NICHT GEBRAUCHT
8ED4 20 D2 F5           JSR  LODOUT   ;'LOADING' AUSGEBEN
               ;
               ;BLOCK LADEN:
               ;
8ED7 A0 00     GL00     LDY  #0
8ED9 20 12 E1  GL0      JSR  BASIN    ;HOLE BYTE
8EDC 91 FD              STA  (USE),Y  ;ABSPEICHERN
8EDE C8                 INY
8EDF C4 AF              CPY  SHP+1
8EE1 D0 F6              BNE  GL0
8EE3 E6 FE              INC  USE+1
8EE5 C6 58              DEC  D1
8EE7 F0 03              BEQ  GL0.     ;LETZTER DURCHGANG
8EE9 10 EE              BPL  GL0
8EEB 60                 RTS
               ;
8EEC A6 57     GL0.     LDX  D0
8EEE 86 AF              STX  SHP+1
8EF0 D0 E7              BNE  GL0      ;UNBEDINGT
               ;
               ;
               ;SUCHE NACH FREIER FILENUMMER:
               ;*****************************
               ;
8EF2 E6 B8     SRCHFN   INC  FILENR
8EF4 F0 FC              BEQ  SRCHFN   ;NICHT 0
8EF6 A6 B8              LDX  FILENR
8EF8 20 0F F3           JSR  $F30F    ;SUCHT FILENUMMER
8EFB F0 F5              BEQ  SRCHFN   ;SCHON VORHANDEN
8EFD 60                 RTS
```

```
              ;
              ;
              ;
              ;
              ;STRING ALS PARAMETER UEBERNEHMEN:
              ;**********************************
              ;
              ;AUSGABE: ADH: STRINGLAENGE
              ;     $22/$23: STRINGADRESSE
              ;
8EFE 20 FD AE GETSTR   JSR  CHKCOM  ;TESTE AUF KOMMA
8F01 20 9E AD          JSR  FRMEVL  ;AUSDRUCK UEBERNEHMEN
8F04 20 8F AD          JSR  CHKSTR  ;TESTE AUF STRING
8F07 20 82 B7          JSR  $B782   ;STRINGPARAMETER HOLEN (LAENGE NACH Y)
8F0A 84 AD             STY  ADH     ;LAENGE(=0 => SPRITE AUS)
8F0C 60                RTS
              ;
              ;
              ;STRING TOO LONG ERROR:
              ;
8F0D 4C 58 B6 ILLFE    JMP  STRERR  ;STRING TOO LONG
              ;
              ;
              ;
              ;***********************
              ;**                   **
              ;**  BEFEHL: SDEFINE  **
              ;**                   **
              ;***********************
              ;
              ;
8F10 20 9E B7 DEFINE   JSR  GETBYT  ;SPRITENUMMER NACH X
8F13 8A                TXA
8F14 29 0F             AND  #15     ;KORREKTUR
8F16 85 AC             STA  ADL
8F18 20 FE 8E          JSR  GETSTR  ;DEFINITIONSSTRING HOLEN
8F1B C0 00             CPY  #0
8F1D F0 5D             BEQ  OFFS    ;LAENGE=0 => SPRITE AUS
8F1F C0 3F             CPY  #63
8F21 90 EA             BCC  ILLFE   ;LAENGE > 63 => ERROR
              ;
```

```
                ;SPRITEDEFINTIONSADRESSE BERECHNEN:
                ;(SPRITENUMMER*64+BASISADRESSE)
                ;
8F23 A5 AC             LDA  ADL     ;SPRITENUMMER
8F25 4A                LSR  A
8F26 4A                LSR  A       ;/4
8F27 18                CLC
8F28 69 D2             ADC  #SPRDEF ;HIGH-BYTE-DEFINITIONSADRESSE
8F2A 85 FE             STA  USE+1
8F2C A5 AC             LDA  ADL     ;SPRITENUMMER
8F2E 29 03             AND  #3      ;NUR UNTERE 2 BITS
8F30 A8                TAY
8F31 B9 51 7D          LDA  STAB2,Y ;MULTIPLIKATIONSTABELLE
8F34 85 FD             STA  USE     ;SPRITEDATENADRESSE
                ;
8F36 78                SEI
8F37 A5 01             LDA  1
8F39 48                PHA
8F3A 29 FC             AND  #$FC
8F3C 85 01             STA  1       ;ROM AUS
                ;
8F3E A0 00             LDY  #0
8F40 B1 22      DE2    LDA  ($22),Y ;ZEICHEN AUS STRING
8F42 91 FD             STA  (USE),Y ;NACH DEFINITIONSBEREICH
8F44 C8                INY
8F45 C4 AD             CPY  ADH
8F47 D0 F7             BNE  DE2
                ;
8F49 68                PLA
8F4A 85 01             STA  1       ;ROM AN
                ;
8F4C A5 AC             LDA  ADL     ;NR.
8F4E AA                TAX          ;ALS INDEX
8F4F A0 01             LDY  #1      ;FLAG ZWEITEN SATZ EINSTELLEN
8F51 18                CLC
8F52 69 48             ADC  #$48    ;DEFINITIONEN LIEGEN AB $48. BLOCK
8F54 9D 19 C7          STA  SPVEK,X ;SPRITEDEFINITIONS-VEKTOREN
                ;
8F57 E0 08             CPX  #8      ;SPRITENUMMER >= 8?
8F59 B0 08             BCS  DE4     ;JA=> SATZ 2
8F5B 88                DEY          ;Y=0/FLAG F. ERSTEN SATZ EINSTELLEN
8F5C 9D F8 CB          STA  $CBF8,X ;SEITE 2 - VEKTORENBUFFER
8F5F 9D F8 C3          STA  $C3F8,X ;AKTUELLER VEKTORENBUFFER
8F62 8A                TXA
8F63 29 07      DE4    AND  #7
8F65 AA                TAX
8F66 B9 88 C6          LDA  SPRIT1,Y  ;HOLE SPRITEENABLE AUF BUFFER (SATZ Y)
8F69 1D 27 7D          ORA  STAB1,X   ;ENABLE SPRITE NR. X
8F6C 99 88 C6 DE2.     STA  SPRIT1,Y  ;WIEDER ZURUECK
8F6F B0 0A             BCS  DE3       ;->SATZ 2 (DA CARRY=1)
8F71 AA                TAX            ;SATZ 1 (DA CARRY=0)
```

```
8F72 A5 02              LDA  GFLAG
8F74 29 08              AND  #8
8F76 F0 03              BEQ  DE3      ;KEINE SPRITES IM TEXTMODUS
8F78 8E 15 D0           STX  V+21     ;SPRITE EINSCHALTEN
8F7B 60        DE3      RTS
               ;
               ;SPRITE AUSSCHALTEN:
               ;
8F7C A5 AC     OFFS     LDA  ADL       ;SPRITENUMMER
8F7E C9 08              CMP  #8
8F80 29 07              AND  #7
8F82 AA                 TAX
8F83 A0 00              LDY  #0
8F85 90 01              BCC  OFFS1     ;->SATZ 1
8F87 C8                 INY            ;SATZ 2
8F88 BD 27 7D OFFS1     LDA  STAB1,X   ;BITMASKE FUER SPRITE
8F8B 49 FF              EOR  #$FF
8F8D 39 88 C6           AND  SPRIT1,Y ;BIT IM SPRITEENABLEBUFFER LOESCHEN
8F90 4C 6C 8F           JMP  DE2.      ;AUSSCHALTEN
               ;
               ;
               ;
               ;*********************
               ;**                 **
               ;**  BEFEHL: SREAD  **
               ;**                 **
               ;*********************
               ;
               ;
8F93 A2 00     SREAD    LDX  #0       ;ZAEHLER BYTES
8F95 86 FD              STX  USE
8F97 A6 41              LDX  $41
8F99 A4 42              LDY  $42      ;DATA ZEIGER
8F9B 86 43              STX  $43
8F9D 84 44              STY  $44      ;INPUT ZEIGER
8F9F 20 8B B0           JSR  $B08B    ;SUCHT VARIABLE
8FA2 85 49              STA  $49
8FA4 84 4A              STY  $4A      ;VAR.-ADRESSE
8FA6 24 0D              BIT  $0D      ;VAR.-TYP
8FA8 30 05              BMI  SR1      ;STRING?
8FAA A2 16              LDX  #22      ;NEIN!
8FAC 4C 37 A4           JMP  $A437    ;TYPE MISMATCH
               ;
8FAF A5 7A     SR1      LDA  PRGZG
8FB1 A4 7B              LDY  PRGZG+1
8FB3 85 4B              STA  $4B
8FB5 84 4C              STY  $4C      ;PROGRAMMZEIGER RETTEN
8FB7 A6 43              LDX  $43
8FB9 A4 44              LDY  $44      ;INPUTZEIGER
8FBB 86 7A              STX  PRGZG
8FBD 84 7B              STY  PRGZG+1 ;ALS PROGRAMMZEIGER
```

```
                 ;
8FBF 20 06 A9 SR2       JSR  $A906   ;NAECHSTES STATEMENT SUCHEN
8FC2 C8                 INY
8FC3 AA                 TAX          ;ZEILENENDE?
8FC4 D0 15              BNE  SR3     ;NEIN
                 ;
8FC6 A2 0D              LDX  #$D
8FC8 C8                 INY
8FC9 B1 7A              LDA  (PRGZG),Y
8FCB D0 03              BNE  SR4     ;NEIN
8FCD 4C 37 A4           JMP  $A437   ;OUT OF DATA
                 ;
8FD0 C8          SR4    INY
8FD1 B1 7A              LDA  (PRGZG),Y  ;ZEILENNUMMER HOLEN
8FD3 85 3F              STA  $3F        ;ALS ZEILENNUMMER FUER DATA
8FD5 C8                 INY
8FD6 B1 7A              LDA  (PRGZG),Y
8FD8 C8                 INY
8FD9 85 40              STA  $40
                 ;
8FDB 20 FB A8 SR3       JSR  $A8FB   ;NAECHSTES STATEM.
8FDE 20 79 00           JSR  CHRGOT  ;LETZTES ZEICHEN NACH A
8FE1 AA                 TAX
8FE2 E0 83              CPX  #$83    ;TOKEN V. DATA ?
8FE4 D0 D9              BNE  SR2     ;NEIN=>WEITERSUCHEN
8FE6 20 73 00 SR0       JSR  CHRGET  ;DATEN LESEN
8FE9 20 F3 BC           JSR  $BCF3   ;ZIFFERNSTRING NACH FAC (FLIESSKOMMAAKKU)
8FEC 20 1B BC           JSR  $BC1B   ;FAC RUNDEN
8FEF 20 BF B1           JSR  $B1BF   ;NACH INTEGER WANDELN
8FF2 A5 64              LDA  $64     ;WERT ZU HOCH? (HIGH-BYTE)
8FF4 F0 03              BEQ  SR5
8FF6 4C 48 B2           JMP  QERR    ;ILLEGAL QUANTITY
8FF9 A4 FD       SR5    LDY  USE
8FFB A5 65              LDA  $65
8FFD 99 00 02           STA  $200,Y  ;WERT SPEICHERN
9000 C8                 INY
9001 84 FD              STY  USE
9003 C0 3F              CPY  #63     ;63 DATEN LESEN
9005 F0 0C              BEQ  SR6     ;FERTIG
9007 20 79 00           JSR  CHRGOT  ;LETZTES ZEICHEN HOLEN
900A F0 B3              BEQ  SR2     ;ZEILENENDE=>NEUE DATENZ.
900C C9 2C              CMP  #","
900E F0 D6              BEQ  SR0     ;NAECHSTES DATUM
9010 4C 4D AB           JMP  $AB4D   ;FEHLER
9013 A5 7A       SR6    LDA  PRGZG
9015 A4 7B              LDY  PRGZG+1
9017 85 43              STA  $43
9019 84 44              STY  $44
901B A5 4B              LDA  $4B
901D A4 4C              LDY  $4C
901F 85 7A              STA  PRGZG
```

```
9021 84 7B              STY  PRGZG+1 ;PROGRAMMZEIGER WIEDERHERSTELLEN
9023 20 79 00           JSR  CHRGOT
9026 F0 03              BEQ  SR7     ;ZEILENENDE
9028 4C 08 AF           JMP  SERR    ;SYNTAX ERROR
902B A5 43     SR7      LDA  $43
902D A4 44              LDY  $44
902F 20 27 A8           JSR  $A827   ;DATA ZEIGER SETZEN
9032 A0 02              LDY  #2
9034 A2 00              LDX  #0      ;STARTADR. STRING =$200
9036 86 62              STX  $62
9038 84 63              STY  $63
903A 86 6F              STX  $6F
903C 84 70              STY  $70
903E 84 72              STY  $72
9040 A0 3F              LDY  #63     ;LAENGE
9042 20 A8 B4           JSR  $B4A8   ;STRING ERZEUGEN
9045 4C 2C AA           JMP  $AA2C   ;ZUWEISUNG AN VARIABLE
               ;
               ;
               ;
               ;*********************
               ;**                 **
               ;**  BEFEHL: SMODE  **
               ;**                 **
               ;*********************
               ;
               ;
9048 20 9E B7 SMODE     JSR  GETBYT  ;SPRITENUMMER->X
904B 8A                 TXA          ;MAX.WERT=15
904C 29 0F              AND  #$0F
904E 85 FD              STA  USE     ;NR.
9050 20 ED 95           JSR  GET15   ;SPRITEMODUS->X
9053 85 FE              STA  USE+1   ;GROESSE+MC+PRIOR.
9055 E0 08              CPX  #8      ;MC:C=1
9057 08                 PHP
9058 20 ED 95           JSR  GET15   ;FARBE 1 -> X
905B 8D 48 C6           STA  ZWIS
905E 28                 PLP          ;MC-FLAG
905F 90 23              BCC  SM2     ;NORMAL MODUS
9061 20 79 00           JSR  CHRGOT
9064 F0 1E              BEQ  SM2     ;KEINE WEITEREN ANGABEN
9066 20 ED 95           JSR  GET15   ;FARBE 2 -> X
9069 85 AD              STA  ADH     ;MC: ;FARBE2
906B 20 ED 95           JSR  GET15   ;FARBE 3 -> X
               ;
               ;FARBEN 2,3 SETZEN:
               ;
906E A6 AD              LDX  ADH
9070 A4 FD              LDY  USE       ;SPRITENUMMER
9072 C0 08              CPY  #8
9074 B0 08              BCS  SM0       ;SATZ 2
```

```
              ;
9076 8D E6 C6          STA  SCLMC1+1  ;SATZ 1
9079 8E E5 C6          STX  SCLMC1    ;MC-FARBEN
907C 90 06             BCC  SM2       ;UNBEDINGT
              ;
907E 8D F0 C6 SM0      STA  SCLMC2+1  ;SATZ 2
9081 8D EF C6          STA  SCLMC2    ;MC-FARBEN
              ;
              ;FARBE 1 SETZEN:
              ;
9084 A6 FD    SM2      LDX  USE       ;NR
9086 AD 48 C6          LDA  ZWIS      ;FARBE 1
9089 E0 08             CPX  #8
908B B0 05             BCS  SM1       ;SATZ 2
908D 9D E7 C6          STA  SCLHG1,X  ;ALS FARBE SP.
9090 90 03             BCC  SM1.      ;UNBED.
              ;
9092 9D E9 C6 SM1      STA  SCLHG2-8,X
              ;
              ;SPRITEMODUS BEARBEITEN:
              ;(INDEXVORBEREITUNG)
              ;
9095 A0 06    SM1.     LDY  #6      ;INDEX 2 (RESULTIERT AUS SPRITESATZ)
9097 8A                TXA          ;INDEX 1 (=SPRITENUMMER)
9098 C9 08             CMP  #8
909A 90 03             BCC  SM5
909C C8                INY          ;INDEX 2 (FUER SATZ 2)
909D 29 07             AND  #7      ;INDEX 1
909F AA       SM5      TAX          ;INDEX 1 NACH X
              ;
              ;SPRITEMODUS BEARBEITEN:
              ;(BITBEARBEITUNG UND PUFFERUNG DER MODUSEINGABE)
              ;
90A0 A9 00    SM6      LDA  #0      ;MASKE=0
90A2 66 FE             ROR  USE+1   ;N.BIT V. MODUS NACH CARRY (X,Y-
GROESSE,ETC.)
90A4 90 02             BCC  SM7
90A6 A9 FF             LDA  #$FF    ;MASKE=$FF
90A8 3D 27 7D SM7      AND  STAB1,X ;ENTSPRECHENDES BIT ISOLIEREN
90AB 85 AC             STA  ADL     ;UND MERKEN
              ;
90AD BD 27 7D          LDA  STAB1,X ;HOLE GEPUFFERTE X-GROESSE
90B0 49 FF             EOR  #$FF
90B2 39 2B C7          AND  SPMC1,Y ;BIT LOESCHEN
90B5 05 AC             ORA  ADL     ;UND NEU FESTLEGEN
90B7 99 2B C7          STA  SPMC1,Y ;ZURUECK NACH BUFFER
90BA 88                DEY
90BB 88                DEY          ;NAECHSTEN MODUSPARAMETER
90BC 10 E2             BPL  SM6
              ;
```

```
              ;REALISIERUNG DER PARAMETER:
              ;(UEBERTRAGUNG: PUFFER->VIC)
              ;
90BE A6 FD              LDX  USE
90C0 E0 08              CPX  #8
90C2 B0 2C              BCS  SM8       ;SATZ ZWEI -> KEINE REALISIERUNG
              ;
90C4 BD E7 C6           LDA  SCLHG1,X ;HGR-FARBEN
90C7 9D 27 D0           STA  V+39,X
90CA A0 00              LDY  #0        ;SATZ EINS
90CC B9 E5 C6 SM9       LDA  SCLMC1,Y ;MC-FARBE (ANST. RAS-IRQ)
90CF 8D 25 D0           STA  V+37
90D2 B9 EF C6           LDA  SCLMC2,Y ;MC-FARBE
90D5 8D 26 D0           STA  V+38
90D8 B9 31 C7           LDA  SPXEX1,Y ;SPRITE X-EXPAND
90DB 8D 1D D0           STA  V+29
90DE B9 2B C7           LDA  SPMC1,Y  ;SPRITE MC
90E1 8D 1C D0           STA  V+28
90E4 B9 2F C7           LDA  SPYEX1,Y ;SPRITE Y-EXPAND
90E7 8D 17 D0           STA  V+23
90EA B9 2D C7           LDA  SPPRI1,Y ;SPRITE-PRIORITAET
90ED 8D 1B D0           STA  V+27
              ;
90F0 60       SM8       RTS
              ;
              ;
              ;
              ;**********************
              ;**                  **
              ;**  BEFEHL: SPOWER  **
              ;**                  **
              ;**********************
              ;
              ;
90F1 20 6C 8A SPRPOW    JSR  GMODEI   ;ALLES AUS
90F4 A5 02              LDA  GFLAG
90F6 29 DF              AND  #%11011111
90F8 85 02              STA  GFLAG    ;BIT 5 =0 (INTERRUPTFLAG)
90FA 20 79 00           JSR  CHRGOT
90FD F0 F1              BEQ  SM8      ;KEINE ZEICHEN MEHR -> FERTIG
              ;
90FF 20 64 8C           JSR  GETRAS   ;STREIFENBREITE HOLEN
9102 A5 02              LDA  GFLAG
9104 29 08              AND  #8
9106 F0 E8              BEQ  SM8      ;TEXT AN=>NICHT ANSCHALTEN
9108 78                 SEI
9109 A2 80              LDX  #$80
910B 8E 94 C6           STX  RASFLG   ;SPRITE-IRQ EIN
910E E8                 INX           ;X=#$81
910F 8E 1A D0           STX  V+26     ;IMR (INTERRUPT MASK REGISTER)
9112 A5 02              LDA  GFLAG
```

```
9114 09 20           ORA  #$20
9116 85 02           STA  GFLAG   ;STATUS SETZEN
9118 58              CLI
9119 60              RTS
              ;
              ;
              ;
              ;**********************
              ;**                  **
              ;**  BEFEHL: PADDLE  **
              ;**                  **
              ;**********************
              ;
              ;
911A 20 8B B0 PDL    JSR  $B08B   ;VARIABLENADRESSE SUCHEN
911D 85 49           STA  $49
911F 84 4A           STY  $4A     ;ZWISCHENSPEICHERN
9121 A5 0E           LDA  $0E
9123 8D AD C6        STA  VARADR  ;REAL/INTEGER-FLAG
9126 A5 0D           LDA  $0D     ;TYPFLAG: NUMERISCH/STRING
9128 F0 03           BEQ  PDL1    ;->NUMERISCH
912A 4C AA 8F        JMP  SR1-5   ;TYP MISMATCH
912D 20 F1 B7 PDL1   JSR  CHKGET  ;PADDELNUMMER HOLEN
9130 CA              DEX          ;AUS 1-4 MACH 0-3
9131 8A              TXA
9132 29 03           AND  #3
9134 4A              LSR  A
9135 4A              LSR  A
9136 A9 80           LDA  #$80    ;PDL A ($80)
9138 90 01           BCC  PDL2    ;PDL A
913A 4A              LSR  A       ;PDL B ($40)
913B 78       PDL2   SEI          ;TASTATUR AUSSER BETRIEB
913C 8D 00 DC        STA  $DC00   ;CIA 1
913F 09 C0           ORA  #$C0    ;AUF AUSGANG SETZEN
9141 8D 02 DC        STA  $DC02
9144 A0 00           LDY  #0
9146 8C 87 C6        STY  ZAHL+1  ;HIGH-BYTE=0
9149 88       PDL3   DEY          ;VERZOEGERUNGSSCHLEIFE ZUR
914A D0 FD           BNE  PDL3    ;BERUHIGUNG D. A/D-EINGANGS
914C 8A              TXA          ;ALTER WERT (PADDELNUMMER)
914D 29 01           AND  #1
914F AA              TAX
9150 BD 19 D4        LDA  S+$19,X ;A/D (1/2)-WERT HOLEN
9153 8D 86 C6        STA  ZAHL    ;LOW-BYTE
9156 A9 FF           LDA  #$FF
9158 8D 02 DC        STA  $DC02   ;ALLE BITS AUF AUSGANG
915B 58              CLI          ;TASTATUR AN
915C 4C CA 7F        JMP  PLTVA.  ;WERT AN VARIABLE
              ;
              ;
              ;
```

```
              ;
              ;ALLE REGISTER RETTEN, DIE
              ;SOWOHL VON DER IRQ- ALS AUCH
              ;VON DER LINIENROUTINE VERWENDET
              ;WERDEN (JEWEILS AUSTAUSCH MIT BUFFER):
              ;****************************************
              ;
              ;
915F A2 14    SREGIN   LDX  #$A+$A  ;FUER INIT-ROUTINE
9161 2C                .BYT $2C     ;NAECHSTEN BEFEHL UEBERSPRINGEN (CODE FUER
BIT)
9162 A2 0E    SREGSG   LDX  #7+7    ;POINTER AUF REGISTER-TABELLE
9164 AC 4E C6          LDY  NR      ;SPRITENUMMER
              ;
9167 A5 01             LDA  1
9169 48                PHA
916A 29 FC             AND  #$FC    ;ROM AUS
916C 85 01             STA  1       ;I-FLAG IST BEREITS 1!
              ;
916E 98                TYA          ;NR
916F 0A                ASL  A
9170 0A                ASL  A
9171 0A                ASL  A
9172 0A                ASL  A       ;*16
9173 A8                TAY
              ;
9174 A5 AC             LDA  ADL
9176 48                PHA
9177 A5 AD             LDA  ADH
9179 48                PHA          ;ADL/ADH RETTEN
              ;
917A E0 0E             CPX  #7+7
917C D0 03             BNE  SRE1    ;ALLE REGISTER DER TABELLE RETTEN
917E C8                INY
917F C8                INY
9180 C8                INY          ;AUCH Y-POINTER ERHOEHEN
              ;
              ;REGISTER DER TABELLE MIT BUFFER TAUSCHEN:
              ;
9181 BD AC 91 SRE1     LDA  REGTAB,X
9184 85 AC             STA  ADL
9186 BD AD 91          LDA  REGTAB+1,X
9189 85 AD             STA  ADH     ;REGISTERADRESSE NACH ADL/ADH
918B 8A                TXA
918C 48                PHA          ;POINTER RETTEN
918D A2 00             LDX  #0
918F A1 AC             LDA  (ADL,X) ;INHALT DES REGISTERS
9191 48                PHA          ;RETTEN
9192 B9 00 D0          LDA  SREG,Y  ;INHALT DES BUFFERS
9195 81 AC             STA  (ADL,X) ;NACH REGISTER
9197 68                PLA          ;ALTER REGISTERINHALT
```

```
9198 99 00 D0          STA  SREG,Y  ;NACH BUFFER (AUSTAUSCH)
919B 68                PLA
919C AA                TAX          ;POINTER NACH X
919D C8                INY
919E CA                DEX
919F CA                DEX          ;NAECHSTES REGISTER
91A0 10 DF             BPL  SRE1
                ;
91A2 68                PLA
91A3 85 AD             STA  ADH
91A5 68                PLA
91A6 85 AC             STA  ADL     ;ADL/ADH WIEDERHOLEN
91A8 68                PLA
91A9 85 01             STA  1       ;ROM EIN (I-FLAG BLEIBT AN!)
91AB 60                RTS
                ;
                ;
                ;TABELLE DER REGISTERADRESSEN:
                ;****************************
                ;
91AC 57 00      REGTAB  .WOR D0
91AE 58 00              .WOR D1
91B0 59 00              .WOR D2
91B2 5A 00              .WOR D3
91B4 5B 00              .WOR D4
91B6 5C 00              .WOR D5
91B8 5F 00              .WOR ZA
91BA 93 00              .WOR E5
91BC 10 C6              .WOR E0
91BE 11 C6              .WOR E1
91C0 12 C6              .WOR E2
                ;
                ;
                ;
                ;********************************
                ;**                            **
                ;**  BEFEHL: IF# ... THEN ...  **
                ;**                            **
                ;********************************
                ;
                ;
                ;TABELLE DER IF#-SECUNDAERBEFEHLE:
                ;*********************************
                ;
91C2 55 44 4C IFZDAT   .ASC "UDLRBC"
                ;
                ;
```

```
                ;BEFEHLSANSTEUERUNG:
                ;********************
                ;
91C8 A2 05      IFZ      LDX  #5        ;AKU ENTHAELT NAECHSTES ZEICHEN
91CA DD C2 91   IFZ1     CMP  IFZDAT,X  ;AKU MIT TABELLE VERGLEICHEN
91CD F0 06               BEQ  IFZ2      ;GEFUNDEN
91CF CA                  DEX
91D0 10 F8               BPL  IFZ1
91D2 4C 08 AF            JMP  SERR      ;NICHT GEFUNDEN->SYNTAX ERROR
                ;
91D5 20 73 00   IFZ2     JSR  CHRGET    ;NAECHSTES ZEICHEN
91D8 E0 05               CPX  #5        ;X=5 => C-SECUNDAERBEFEHL
91DA F0 2C               BEQ  IFZ5      ;C-BEFEHLE
91DC 8A                  TXA
91DD 48                  PHA            ;NUMMER DES SECUNDAERBEFEHLS MERKEN
                ;
                ;JOYSTICKBEFEHLE:
                ;
91DE 20 9E B7            JSR  GETBYT    ;HOLE PARAMETER (PORTNUMMER)
91E1 CA                  DEX            ;AUS 1-2 MACH 0-1
91E2 8A                  TXA
91E3 4A                  LSR  A         ;PORTNUMMER NACH CARRY
91E4 68                  PLA            ;BEFEHLSNUMMER
91E5 AA                  TAX            ;NACH X
                ;
91E6 AD 02 DC            LDA  $DC02
91E9 48                  PHA
91EA A9 E0               LDA  #$E0
91EC 8D 02 DC            STA  $DC02     ;TASTATUR AUS
                ;
91EF BD 27 7D            LDA  STAB1,X   ;ENTSPRECHENDES BIT SETZEN
91F2 90 05               BCC  IFZ3      ;PORT 1
                ;
91F4 2D 00 DC            AND  $DC00     ;PORT 2
91F7 B0 03               BCS  IFZ4      ;UNBEDINGT
                ;
91F9 2D 01 DC   IFZ3     AND  $DC01     ;PORT 1
                ;
91FC 5D 27 7D   IFZ4     EOR  STAB1,X   ;BIT UMDREHEN
91FF 85 61               STA  $61       ;UND ALS ERGEBNIS EINER LOGISCHEN VERKN.
MERKEN
9201 68                  PLA
9202 8D 02 DC            STA  $DC02     ;TASTATUR AN
9205 4C 2B A9            JMP  $A92B     ;BASIC-VERZWEIGUNG GEMAESS $61 AUSFUEHREN
                ;
                ;
```

```
              ;C-BEFEHLE:
              ;
9208 C9 43    IFZ5     CMP  #"C"
920A D0 07             BNE  IFZ6    ;IF# C... ODER IF# CN...
              ;
920C A2 00             LDX  #0      ;IF# CC
920E 8E 8B C6          STX  CFLAG   ;KEIN INTERRUPT MEHR
9211 F0 04             BEQ  IFZ7    ;UNBEDINGT
              ;
              ;
9213 C9 4E    IFZ6     CMP  #"N"
9215 D0 11             BNE  IFZ8    ;IF# C...
              ;
              ;IF# CN...:
              ;
9217 AD 1E D0 IFZ7     LDA  V+30
921A 0D 1F D0          ORA  V+31    ;KOLLISIONSREGISTER
921D 85 61             STA  $61     ;ERFUELLT-FLAG FUER BASIC-VERZWEIGUNG
921F 20 E1 92          JSR  CLRCOL  ;KOLLISIONSREGISTER LOESCHEN
9222 20 73 00          JSR  CHRGET
9225 4C 2B A9          JMP  $A92B   ;BASIC-VERZWEIGUNG GEMAESS $61
              ;
              ;
              ;IF# C...:
              ;
9228 A2 80    IFZ8     LDX  #$80    ;FLAG SETZEN FUER:
922A 8E 8B C6          STX  CFLAG   ;"KOLLISION UEBERPRUEFEN"
922D A5 39             LDA  $39
922F 8D 8C C6          STA  ZLNR
9232 A5 3A             LDA  $3A
9234 8D 8D C6          STA  ZLNR+1  ;ZEILENNUMMER MERKEN
9237 A5 7A             LDA  $7A
9239 8D 8E C6          STA  PRGZGR
923C A5 7B             LDA  $7B
923E 8D 8F C6          STA  PRGZGR+1  ;PROGRAMMZEIGER MERKEN
9241 20 E1 92          JSR  CLRCOL    ;VORHERIGE COLL. LOESCHEN
9244 4C 3B A9          JMP  $A93B     ;NAECHSTE ZEILE
              ;
              ;
              ;
```

```
              ;HOLE SPRITENUMMER (IN X) NACH NR
              ;UND SATZNUMMER NACH Y:
              ;*********************************
              ;
9247 8A       GETNR   TXA          ;SPRITENUMMER
9248 29 0F            AND  #$0F    ;16 SPRITES MAX.
924A 8D 4E C6         STA  NR
924D A0 00            LDY  #0      ;SATZ 1 => Y=0
924F 29 07            AND  #$07
9251 CD 4E C6         CMP  NR
9254 F0 01            BEQ  GET1
9256 C8               INY          ;SATZ 2 => Y=1
9257 60       GET1    RTS
              ;
              ;
              ;
              ;********************
              ;**                **
              ;**  BEFEHL: SSET  **
              ;**                **
              ;********************
              ;
              ;
9258 20 9E B7 SSET    JSR  GETBYT   ;HOLE SPRITENUMMER -> X
925B 78       SS7     SEI
925C 20 47 92         JSR  GETNR    ;X->NR ETC.
925F BD 27 7D         LDA  STAB1,X
9262 39 6E C6         AND  SRUNS1,Y ;MERKER FUER SICH BEWEGENDE SPRITES
9265 F0 05            BEQ  SS1      ;SPRITE IST NICHT IN BEWEGUNG
9267 20 6C 93         JSR  SWAITI   ;INTERNE SPRITEWAIT-ROUT.=>WARTEN BIS
SPRITE STOPPT
926A F0 EF            BEQ  SS7      ;UNBED.
              ;
              ;SPRITE AUF SPRITECURSOR SETZEN:
              ;
926C 20 5F 91 SS1     JSR  SREGIN  ;HOLE SPRITE- UND RETTE HLINE-REGISTER
926F AD 10 C6         LDA  E0      ;X-KOORDINATE (LOW-BYTE)
9272 AE 11 C6         LDX  E1      ;X-KOORDINATE (HIGH-BYTE)
9275 AC 12 C6         LDY  E2      ;Y-KOORDINATE
9278 20 17 93         JSR  SSTCR2  ;SPRITE POSITIONIEREN
927B 20 79 00         JSR  CHRGOT  ;LETZTES ZEICHEN HOLEN
              ;
927E C9 A4            CMP  #TO
9280 F0 10            BEQ  SS0     ;DIREKT SSET TO...
              ;
```

```
               ;SPRITEKOORDINATEN X1,Y1 HOLEN:
               ;
9282 20 FD AE            JSR  CHKCOM  ;AUF KOMMA TESTEN
9285 20 EE 92            JSR  STEST   ;KOORD. HOLEN + TESTEN
9288 20 0E 93            JSR  SSTCOR  ;SPEICHERE SPRITE-KOORD. UND POSITIONIERE
928B 20 79 00            JSR  CHRGOT  ;LETZTES ZEICHEN
928E C9 A4               CMP  #TO
9290 D0 37               BNE  SS9A    ;KEIN TO
               ;
               ;SPRITEZIELKOORDINATEN HOLEN
               ;UND SPRITEBEWEGUNG INITIALISIEREN:
               ;
9292 A9 42     SS0       LDA  #$42
9294 05 02               ORA  GFLAG
9296 85 02               STA  GFLAG   ;AUSSTEIGERFLAG-SPRITE
9298 20 7C 93            JSR  SLDCOR  ;LADE SPRITEKOORDINATEN->SX/SY
929B 20 73 00            JSR  CHRGET
929E 20 EE 92            JSR  STEST   ;ZIELKOORDINATEN HOLEN
92A1 20 E4 87            JSR  HLINE   ;LINIENINITIALISIERUNG (BEACHTE
AUSSTEIGERFLAG!)
92A4 F0 20               BEQ  SS9     ;BEWEGUNG ZUENDE
               ;
92A6 08                  PHP          ;FLAGS RETTEN
92A7 AC 4E C6            LDY  NR      ;SPRITENUMMER
92AA 8A                  TXA
92AB 99 76 C6            STA  X.SREG,Y  ;X-REGISTER RETTEN
92AE 68                  PLA
92AF 99 F9 C6            STA  PFLAGS,Y  ;CARRY RETTEN (I=1!)
92B2 98                  TYA            ;NR.
92B3 C0 08               CPY  #8
92B5 29 07               AND  #7
92B7 A8                  TAY
92B8 B9 27 7D            LDA  STAB1,Y   ;SPRITEBIT SETZEN
92BB A0 00               LDY  #0        ;SATZ 1
92BD 90 01               BCC  SS3
92BF C8                  INY            ;SATZ 2
92C0 19 6E C6 SS3        ORA  SRUNS1,Y  ;SPRITE IN BEWEGUNG!
92C3 99 6E C6            STA  SRUNS1,Y  ;ALS IN BEWEGUNG KENNZEICHNEN
               ;
92C6 20 02 93 SS9        JSR  SSTCR1    ;KOORD. ZUR BEWEGUNG
92C9 20 5F 91 SS9A       JSR  SREGIN    ;REGISTER WIEDER WECHSELN
92CC 20 79 00            JSR  CHRGOT    ;LETZTES ZEICHEN HOLEN
92CF F0 0A               BEQ  SS8       ;BEFEHL ZUENDE
               ;
               ;GESCHWINDIGKEIT HOLEN:
               ;
92D1 20 F1 B7            JSR  CHKGET    ;GESCHWINDIGKEIT->X
92D4 8A                  TXA
92D5 AC 4E C6            LDY  NR
92D8 99 09 C7            STA  SSCHRT,Y  ;ANZAHL SCHRITTE/IRQ
               ;
```

```
              ;TERMINIERUNG:
              ;
92DB A5 02    SS8      LDA  GFLAG
92DD 29 BD             AND  #%10111101
92DF 85 02             STA  GFLAG     ;AUSSTEIGERFLAG LOESCHEN
              ;
              ;KOLLSISIONEN AUSSCHALTEN:
              ;
92E1 A9 00    CLRCOL   LDA  #0
92E3 8D 1E D0          STA  V+30
92E6 8D 1F D0          STA  V+31
92E9 8D 8A C6          STA  IRRSAV
92EC 58                CLI
92ED 60                RTS
              ;
              ;
              ;
              ;HOLE SPRITEKOORDINATEN:
              ;(ALS PARAMETER BEI SPRITEBEFEHLEN)
              ;***********************************
              ;
              ;AUSGABE:
              ;        A: X-KOORDINATE (LOW)
              ;        X: X-KOORDINATE (HIGH)
              ;        Y: Y-KOORDINATE
              ;
              ;
92EE A5 02    STEST    LDA  GFLAG
92F0 48                PHA
92F1 09 04             ORA  #4      ;FLAG FUER
92F3 85 02             STA  GFLAG   ;NICHT TESTEN
92F5 20 FF 7F          JSR  TESCOR
92F8 68                PLA
92F9 85 02             STA  GFLAG   ;ALTES GFLAG WIEDERHERSTELLEN
92FB 8A                TXA
92FC 29 01             AND  #1      ;KORRIGIEREN
92FE AA                TAX
92FF A5 14             LDA  XK
9301 60                RTS
              ;
              ;
              ;KOORDINATEN IN BUFFER SPEICHERN
              ;UND SPRITE POSITIONIEREN:
              ;********************************
              ;
9302 AD 39 C6 SSTCR1   LDA  SXL
9305 AE 3A C6          LDX  SXH
9308 AC 3B C6          LDY  SY      ;KOORDINATEN IN REGISTER
930B 4C 17 93          JMP  SSTCR2
              ;
```

```
930E 8D 10 C6 SSTCOR    STA  E0        ;STORE SPRITE KOORD.
9311 8E 11 C6           STX  E1
9314 8C 12 C6           STY  E2        ;KOORDINATEN IN REGISTER
              ;
              ;
              ;EINGABE:
              ;         A: X-KOORDINATE (LOW)
              ;         X: X-KOORDINATE (HIGH)
              ;         Y: Y-KOORDINATE
              ;
9317 48       SSTCR2    PHA            ;X-K-LOW
9318 98                 TYA            ;Y-K
9319 48                 PHA            ;RETTEN
931A AD 4E C6           LDA  NR        ;NR       ACHTUNG! I-FLAG=1
931D 0A                 ASL  A         ;*2
931E A8                 TAY            ;NACH Y
931F C9 10              CMP  #8+8      ;>=16?
9321 90 01              BCC  SST2      ;NEIN
9323 C8                 INY            ;JA => UEBERSPRINGE XK-HIGH
9324 68       SST2      PLA                 ;Y-K
9325 99 C5 C6           STA  SKORD1+1,Y  ;IN BUFFER
9328 B0 03              BCS  SST3           ;2.SATZ
932A 99 01 D0           STA  V+1,Y          ;NUR NR. UNTER 8 IN VIC
932D 68       SST3      PLA                 ;X-K-LOW
932E 99 C4 C6           STA  SKORD1,Y       ;IN BUFFER
9331 B0 03              BCS  SST4           ;SATZ 2
9333 99 00 D0           STA  V,Y            ;NUR BEI NR. UNTER 8 IN VIC
              ;
              ;X-KOORDINATE HIGH-BYTE:
              ;
9336 AD 4E C6 SST4      LDA  NR
9339 29 07              AND  #$7
933B A8                 TAY            ;SPRITENUMMER->Y
933C 8A                 TXA            ;X-K-HIGH
933D F0 02              BEQ  SST1      ;NULL
933F A9 FF              LDA  #$FF
9341 39 27 7D SST1      AND  STAB1,Y ;ENTSPRECHENDES BIT ANSPRECHEN
9344 8D AC C6           STA  ZWISIQ  ;ZWISCHENSPEICHERN
9347 B9 27 7D           LDA  STAB1,Y
934A 49 FF              EOR  #$FF
934C B0 0E              BCS  SST5
934E 2D D4 C6           AND  SKORD1+16  ;SATZ 1 / BIT AUSBLENDEN
9351 0D AC C6           ORA  ZWISIQ
9354 8D D4 C6           STA  SKORD1+16  ;UND NEU BELEGEN
9357 8D 10 D0           STA  V+16       ;REALISIEREN
935A 90 09              BCC  SST6       ;UNBEDINGT
              ;
935C 2D E5 C6 SST5      AND  SKORD2+16
935F 0D AC C6           ORA  ZWISIQ
9362 8D E5 C6           STA  SKORD2+16  ;WIE OBEN
9365 60       SST6      RTS
```

```
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: SWAIT  **
              ;**                 **
              ;*********************
              ;
              ;
9366 20 9E B7 SWAIT    JSR  GETBYT  ;SPRITENUMMER->X
9369 8E 4E C6          STX  NR      ;SPEICHERN
              ;
              ;INTERNE SPRITEWARTEROUTINE:
              ;
936C AE 4E C6 SWAITI   LDX  NR      ;NR
936F 20 47 92          JSR  GETNR   ;NACH NR ETC.
9372 58                CLI
              ;
9373 BD 27 7D SWAIT1   LDA  STAB1,X
9376 39 6E C6          AND  SRUNS1,Y ;MERKER TESTEN
9379 D0 F8             BNE  SWAIT1   ;WARTEN BIS IRQ-ROUTINE BIT LOESCHT
937B 60                RTS           ;SPRITE STEHT
              ;
              ;
              ;
              ;LADE SPRITEKOORDINATEN AUS BUFFER:
              ;************************************
              ;
937C AD 4E C6 SLDCOR   LDA  NR      ;NR.
937F 48                PHA          ;MERKEN
9380 C9 08             CMP  #8      ;SATZNUMMER?
9382 29 07             AND  #7
9384 A8                TAY
              ;
              ;X-KOORDINATE (HIGH-BYTE):
              ;
9385 B9 27 7D          LDA  STAB1,Y   ;HOLE ENTSPRECHENDES BIT
9388 90 05             BCC  SLD2      ;SATZ 1
938A 2D E5 C6          AND  SKORD2+16 ;X-K-HIGH
938D B0 03             BCS  SLD3      ;UNBED.
              ;
938F 2D D4 C6 SLD2     AND  SKORD1+16 ;X-K-H
9392 F0 02    SLD3     BEQ  SLD1      ;=0
9394 A9 01             LDA  #1        ;=1
9396 8D 3A C6 SLD1     STA  SXH       ;    ACHTUNG! I-FLAG=1
              ;
```

```
              ;X- (LOW-BYTE) UND Y-KOORDINATEN:
              ;
9399 68                PLA
939A 0A                ASL  A         ;NR.*2
939B A8                TAY            ;->Y
939C C0 10             CPY  #8+8
939E 90 01             BCC  SLD4
93A0 C8                INY             ;SATZ 2
93A1 B9 C5 C6 SLD4     LDA  SKORD1+1,Y ;KOORDINATE HOLEN
93A4 8D 3B C6          STA  SY         ;SPEICHERN
93A7 B9 C4 C6          LDA  SKORD1,Y   ;X-KOORD. LOW
93AA 8D 39 C6          STA  SXL        ;SPEICHERN
93AD 60                RTS
              ;
              ;
              ;SPRITE-SATZ 1 AKTIVIEREN:
              ;**************************
              ;
93AE AD 88 C6 SOFF16   LDA  SPRIT1
93B1 8D 15 D0          STA  V+21    ;SP-ENABLE
93B4 A0 00             LDY  #0      ;SATZ 1
93B6 20 CC 90          JSR  SM9     ;MC-FARBEN/MC/X-Y-EXP/PRIOR
93B9 A2 07             LDX  #7
              ;
93BB BD C4 C6 SOFF17   LDA  SKORD1,X
93BE 9D 00 D0          STA  V,X     ;KOORD.
93C1 BD CC C6          LDA  SKORD1+8,X
93C4 9D 08 D0          STA  V+8,X   ;KOORD.(OBERE 4 PAARE)
93C7 BD E7 C6          LDA  SCLHG1,X
93CA 9D 27 D0          STA  V+39,X  ;HGR-FARBEN
93CD BD 19 C7          LDA  SPVEK,X
93D0 9D F8 C3          STA  $C3F8,X ;SPRITEVEKTOREN (SEITE1)
93D3 CA                DEX
93D4 10 E5             BPL  SOFF17
93D6 60                RTS
              ;
              ;
              ;
              ;
              ;**************
              ;*IRQ-ROUTINEN*
              ;**************
              ;
```

```
              ;RASTERZEILEN-IRQ:
              ;*****************
              ;
93D7 2C 94 C6 IRQNEU   BIT  RASFLG  ;RASTER-IRQ AN?
93DA 10 14             BPL  SIRQGO  ;NEIN!
              ;
93DC AD 19 D0          LDA  V+25
93DF 8D 19 D0          STA  V+25    ;IRR (INTERRUPT REQUEST REGISTER) LOESCHEN
93E2 30 0F             BMI  IRQ0    ;IRQ V. RASTER?
93E4 AD 0D DC          LDA  $DC0D   ;CIA 2 - IRR LOESCHEN
93E7 AD 6E C6          LDA  SRUNS1
93EA 0D 6F C6          ORA  SRUNS2  ;SPRITE IN BEWEGUNG?
93ED D0 01             BNE  SIRQGO  ;JA => KEIN RAS-IRQ! (ZEITPROBLEM!)
93EF 58                CLI          ;RAS-IRQ WAEHREND ALT-IRQ ERLAUBEN
93F0 4C 69 94 SIRQGO   JMP  SIRQ    ;NORMALEN IRQ BEDIENEN
              ;
93F3 A2 03    IRQ0     LDX  #3
93F5 CA       IRQWAR   DEX
93F6 D0 FD             BNE  IRQWAR  ;WARTEN
              ;
93F8 2C 94 C6          BIT  RASFLG  ;SPRITE-/TEXT-IRQ?
93FB 50 22             BVC  RASSPR  ;SPRITE-RASTER-IRQ
93FD AD 95 C6          LDA  RSFLG1
9400 49 80             EOR  #$80
9402 8D 95 C6          STA  RSFLG1  ;OBEN/UNTEN-FLAG
9405 30 0C             BMI  IRQ1    ;UNTEN
9407 AD 38 C6          LDA  RAST2   ;ZEILENNUMMER OBEN
940A 8D 12 D0          STA  V+18    ;NEUE IRQ-ZEILE
940D 20 4F 8B          JSR  TEXTM   ;TEXT UMSCHALTEN
9410 4C BC FE          JMP  $FEBC   ;ABSCHLIESSEN
              ;
9413 AD 37 C6 IRQ1     LDA  RAST1   ;ZEILENNUMMER UNTEN
9416 8D 12 D0          STA  V+18    ;NEUE IRQ-ZEILE
9419 20 FD 8A          JSR  GRAPM1  ;GRAPHIK EINSCHALTEN
941C 4C BC FE          JMP  $FEBC   ;ABSCHLIESSEN
              ;
941F AD 95 C6 RASSPR   LDA  RSFLG1
9422 49 80             EOR  #$80    ;OBEN/UNTEN-FLAG WECHSELN
9424 8D 95 C6          STA  RSFLG1
9427 30 31             BMI  IRQ2    ;UNTEN
              ;
9429 AD 38 C6          LDA  RAST2
942C 8D 12 D0          STA  V+18    ;NEUE IRQ-ZEILE
942F AD 89 C6          LDA  SPRIT2  ;OBEN
9432 8D 15 D0          STA  V+21    ;SP-ENABLE
9435 A0 01             LDY  #1      ;SATZ 2 AKTIVIEREN
9437 20 CC 90          JSR  SM9     ;MC-FARBEN/MC/X-Y-EXP/PRIOR
943A A2 07             LDX  #7
943C BD D5 C6 IRQ3     LDA  SKORD2,X
943F 9D 00 D0          STA  V,X     ;KOORD. UEBERTRAGEN
9442 BD DD C6          LDA  SKORD2+8,X
```

```
9445 9D 08 D0          STA  V+8,X    ;KOORD. UEBERTRAGEN
9448 BD F1 C6          LDA  SCLHG2,X
944B 9D 27 D0          STA  V+39,X  ;HGR-FARBEN
944E BD 21 C7          LDA  SPVEK+8,X
9451 9D F8 C3          STA  $C3F8,X ;SP-VEKTOREN (SEITE1)
9454 CA                DEX
9455 10 E5             BPL  IRQ3
9457 4C BC FE          JMP  $FEBC    ;ABSCHLIESSEN
              ;
945A AD 37 C6 IRQ2     LDA  RAST1
945D 8D 12 D0          STA  V+18     ;NEUE IRQ-ZEILE
9460 20 AE 93          JSR  SOFF16  ;SATZ 1 EINSCHALTEN
9463 4C BC FE          JMP  $FEBC    ;ABSCHLIESSEN
              ;
              ;IRQ-ROUTINE F. SPRITEBEWEGUNG:
              ;
9466 4C 06 95 JUMPTI   JMP  TONIRQ  ;ZUM TON-IRQ
              ;
9469 AD 6E C6 SIRQ     LDA  SRUNS1  ;MERKER
946C 0D 6F C6          ORA  SRUNS2  ;MERKER2
946F F0 F5             BEQ  JUMPTI  ;KEINE SPRITES IN BEWEGUNG
9471 CE 30 C6          DEC  SLOW     ;ZAEHLER DECREMIEREN
9474 10 F0             BPL  JUMPTI
              ;
9476 A2 03             LDX  #3       ;ZAEHLER WIEDER 'RAUFSETZEN
9478 8E 30 C6          STX  SLOW
947B AD 6E C6          LDA  SRUNS1
947E 48                PHA
947F AD 6F C6          LDA  SRUNS2
9482 48                PHA           ;LAUFMERKER RETTEN
9483 A9 FF             LDA  #$FF
9485 8D 4E C6          STA  NR       ;ZAEHLER=0
9488 68       SI1      PLA           ;LAUFMERKER SATZ 2
9489 AA                TAX
948A D0 04             BNE  SI2      ;IN BETRIEB
948C 68                PLA           ;MERKER 1
948D F0 71             BEQ  SI0      ;KEIN WEITERES IN BETRIEB
948F 48                PHA
9490 8A       SI2      TXA           ;MERKER 2
9491 EE 4E C6          INC  NR       ;INC ZAEHLER (NR.)
9494 4A                LSR  A        ;BIT FUER BIT UEBERPRUEFEN
9495 AA                TAX           ;ZURUECK
9496 68                PLA           ;MERKER 1
9497 6A                ROR  A        ;AUCH NACH RECHTS BIT NACH CARRY
9498 48                PHA           ;ZURUECK
9499 8A                TXA
949A 48                PHA           ;NAECHSTES SPRITE
949B 90 EB             BCC  SI1      ;NICHT IN BEWEGUNG
              ;
```

```
                ;ENTSPRECHENDES SPRITE BEWEGEN:
                ;
949D 20 62 91           JSR  SREGSG  ;HOLE SPRITEREG.+RETTE HLINE
94A0 A5 02              LDA  GFLAG
94A2 09 42              ORA  #$42
94A4 85 02              STA  GFLAG   ;SPRITE-AUSSTEIGERFLAG
94A6 20 7C 93           JSR  SLDCOR  ;LADE SPRITE-KOORDINATEN
94A9 AC 4E C6           LDY  NR
94AC B9 09 C7           LDA  SSCHRT,Y  ;ANZAHL SCHRITTE
94AF 8D AC C6           STA  ZWISIQ    ;MERKEN
94B2 BE 76 C6           LDX  X.SREG,Y  ;X-BUFFER
94B5 B9 F9 C6           LDA  PFLAGS,Y  ;UND FLAGS HOLEN
94B8 48                 PHA
94B9 28                 PLP
94BA 20 47 88 SI7       JSR  L1.     ;SPRITE BEWEGEN
94BD D0 1C    SI3       BNE  SI6     ;BEWEGUNG ZUENDE?
                ;
                ;JA:
                ;
94BF E8                 INX
94C0 8E AC C6           STX  ZWISIQ
94C3 AD 4E C6           LDA  NR      ;NR.
94C6 C9 08              CMP  #8
94C8 29 07              AND  #7
94CA AA                 TAX
94CB BD 27 7D           LDA  STAB1,X
94CE 49 FF              EOR  #$FF
94D0 A2 00              LDX  #0
94D2 90 01              BCC  SI5
94D4 E8                 INX            ;SATZ 2
94D5 3D 6E C6 SI5       AND  SRUNS1,X  ;LAUFMERKER
94D8 9D 6E C6           STA  SRUNS1,X  ;SPRITE LAEUFT NICHT
                ;
94DB AD 1E D0 SI6       LDA  V+30
94DE 0D 1F D0           ORA  V+31    ;KOLLISIONSREGISTER
94E1 F0 03              BEQ  SI4     ;KEINE KOLLISION
94E3 8D 8A C6           STA  IRRSAV  ;KOLLISION
94E6 CE AC C6 SI4       DEC  ZWISIQ  ;ZAEHLER
94E9 D0 CF              BNE  SI7
                ;
94EB 08                 PHP
94EC AC 4E C6           LDY  NR
94EF 8A                 TXA
94F0 99 76 C6           STA  X.SREG,Y
94F3 68                 PLA
94F4 99 F9 C6           STA  PFLAGS,Y  ;LETZTE REGISTER RETTEN
94F7 20 02 93           JSR  SSTCR1    ;KOORD. ZUR BEWEGUNG
94FA 20 62 91           JSR  SREGSG    ;REG. ZURUECK+HLINE HOLEN
94FD 4C 88 94           JMP  SI1       ;NAECHSTES SPRITE
                ;
                ;
```

```
9500 A5 02      SIO      LDA  GFLAG
9502 29 BD               AND  #%10111101
9504 85 02               STA  GFLAG      ;IRQ-FLAG LOESCHEN
                ;
                ;
                ;
                ;
                ;TON-IRQ:
                ;
                ;
9506 A2 03      TONIRQ   LDX  #3        ;STIMMENZAEHLER
9508 CA         TO1      DEX
9509 30 2A               BMI  TOO       ;FERTIG
                ;
950B BD 71 C6            LDA  WELLFO,X  ;WELLENFORMBUFFER
950E 4A                  LSR  A         ;BIT 0 -> STIMME AN?
950F 90 F7               BCC  TO1       ;NEIN
                ;JA:
9511 BD 31 C6            LDA  TIMEL,X
9514 E9 01               SBC  #1        ;C=1
9516 9D 31 C6            STA  TIMEL,X
9519 B0 ED               BCS  TO1
951B BD 34 C6            LDA  TIMEH,X
951E E9 00               SBC  #0        ;TIMER ERNIEDRIGEN
9520 9D 34 C6            STA  TIMEH,X
9523 B0 E3               BCS  TO1       ;NOCH NICHT AUF 0
                ;
9525 BD 71 C6            LDA  WELLFO,X
9528 29 FE               AND  #$FE
952A 9D 71 C6            STA  WELLFO,X  ;TIMER AUF 0->STIMME AUS
952D BC F0 96            LDY  POITAB,X  ;STIMME*7
9530 99 04 D4            STA  S+4,Y     ;STIMME AUSSCHALTEN
9533 90 D3               BCC  TO1       ;UNBEDINGT
                ;
                ;
9535 6C 2A C6   TOO      JMP  (IRQSAV)  ;EIGENDL. IRQ-ROUT.
                ;
                ;
                ;
```

```
              ;**********************
              ;**                  **
              ;**  BEFEHL: VOLUME  **
              ;**                  **
              ;**********************
              ;
              ;
9538 20 9E B7 VOLUME   JSR  GETBYT   ;LAUTSTAERKE->X
953B 20 F0 95          JSR  GET      ;15ER-ZAHL
953E 85 FE             STA  USE+1
9540 AD 74 C6          LDA  VOLUM    ;LAUTSTAERKE-BUFFER
9543 29 F0             AND  #$F0
9545 05 FE    VO1      ORA  USE+1
9547 8D 18 D4          STA  S+24     ;LAUTSTAERKE REALISIEREN
954A 8D 74 C6          STA  VOLUM    ;UND IN BUFFER
954D 60                RTS
              ;
              ;
              ;
              ;*********************
              ;**                 **
              ;**  BEFEHL: SOUND  **
              ;**                 **
              ;*********************
              ;
              ;
954E 20 F4 95 SOUND    JSR  STIMME   ;STIMME NACH ZWIS UND X
9551 86 FE             STX  USE+1
              ;
9553 20 ED 95          JSR  GET15    ;WELLENFORM NACH A
9556 C9 09             CMP  #9       ;WELLENFORM:0-8
9558 90 02             BCC  XXX
955A A9 01             LDA  #1       ;KORREKTUR
955C 48       XXX      PHA           ;MERKEN
              ;
              ;ATTACK, DECAY, SUSTAIN, RELEASE HOLEN:
              ;
955D A0 04             LDY  #4
955F 84 FD             STY  USE      ;ZAEHLER FUER 4 PARAMETER
9561 20 ED 95 SOU1     JSR  GET15    ;PARAMETER NACH A
9564 48                PHA           ;MERKEN
9565 C6 FD             DEC  USE      ;ATT.,DEC.,SUST.,REL.
9567 D0 F8             BNE  SOU1     ;NAECHSTEN
              ;
9569 A6 FE             LDX  USE+1      ;POINTER ERSTELLEN
956B BC F0 96          LDY  POITAB,X  ;STIMME*7
956E 8C 48 C6          STY  ZWIS       ;+ MERKEN
9571 20 79 00          JSR  CHRGOT   ;LETZTES ZEICHEN
9574 F0 3E             BEQ  SOU3     ;BEFEHL ENDE
              ;
```

```
              ;FILTERZUSTAND HOLEN:
              ;
9576 20 F1 B7          JSR  CHKGET  ;PARAMETER NACH X
9579 8A                TXA          ;FILTER ON/OFF
957A 4A                LSR  A       ;0=OFF/1=ON
957B A9 00             LDA  #0      ;KORREKTUR
957D A6 FE             LDX  USE+1   ;STIMME->X
957F 2A                ROL  A
9580 F0 02             BEQ  SOU2
9582 A9 07             LDA  #7
9584 3D 27 7D SOU2     AND  STAB1,X ;MASKE
9587 85 FD             STA  USE
9589 BD 27 7D          LDA  STAB1,X
958C 49 FF             EOR  #$FF
958E 2D 70 C6          AND  FILON   ;FILTERSCHALTER
9591 05 FD             ORA  USE
9593 8D 17 D4          STA  S+23    ;ON/OFF REALISIEREN
9596 8D 70 C6          STA  FILON   ;ON/OFF IN BUFFER
              ;
9599 20 79 00          JSR  CHRGOT
959C F0 16             BEQ  SOU3    ;KEIN ZEICHEN MEHR
              ;
              ;PULSRATE HOLEN:
              ;
959E 20 FD AE          JSR  CHKCOM  ;AUF KOMMA PRUEFEN
95A1 20 8A AD          JSR  $AD8A   ;FRMNUM HOLT NUMER.WERT
95A4 20 F7 B7          JSR  $B7F7   ;FAC NACH ADR.FORMAT
95A7 A5 14             LDA  XK      ;PULSRATE
95A9 AE 48 C6          LDX  ZWIS    ;STIMME
95AC 9D 02 D4          STA  S+2,X   ;REALISIEREN
95AF A5 15             LDA  XK+1
95B1 9D 03 D4          STA  S+3,X   ;HIGH-BYTE
              ;
              ;HUELLKURVE REALISIEREN:
              ;
95B4 AE 48 C6 SOU3     LDX  ZWIS    ;STIMME
95B7 A0 04             LDY  #4
95B9 84 FD             STY  USE
95BB 68       SOU5     PLA          ;WERT VOM STACK
              ;
95BC A0 04             LDY  #4
95BE 4A       SOU6     LSR  A
95BF 66 AC             ROR  ADL
95C1 66 AD             ROR  ADH     ;SETZE ATTACK,DECAY,...
95C3 88                DEY
95C4 D0 F8             BNE  SOU6    ;NACH ADL/ADH ROLLEN
              ;
95C6 C6 FD             DEC  USE     ;NAECHSTEN WERT
95C8 D0 F1             BNE  SOU5
              ;
```

```
95CA A5 AC              LDA  ADL
95CC 9D 05 D4           STA  S+5,X
95CF A5 AD              LDA  ADH
95D1 9D 06 D4           STA  S+6,X   ;UND REALISIEREN
                 ;
95D4 A4 FE              LDY  USE+1   ;STIMME
95D6 68                 PLA          ;WELLENFORM-BITWEISE
95D7 0A                 ASL  A
95D8 0A                 ASL  A
95D9 0A                 ASL  A
95DA 0A                 ASL  A       ;INS OBERE NIBBLE
95DB 78                 SEI          ;WEGEN TON-IRQ
95DC 85 FD              STA  USE
95DE B9 71 C6           LDA  WELLFO,Y
95E1 29 0F              AND  #$0F
95E3 05 FD              ORA  USE
95E5 9D 04 D4           STA  S+4,X     ;WELLENFORM SETZEN
95E8 99 71 C6           STA  WELLFO,Y  ;UND IN PUFFER
95EB 58                 CLI
95EC 60                 RTS
                 ;
                 ;
                 ;4-BIT-PARAMETER HOLEN:
                 ;**********************
                 ;
95ED 20 F1 B7 GET15     JSR  CHKGET  ;PARAMETER->X
95F0 8A       GET       TXA
95F1 29 0F              AND  #$0F    ;OBERES NIBBLE WEG
95F3 60                 RTS
                 ;
                 ;
                 ;STIMME HOLEN:
                 ;*************
                 ;
95F4 20 9E B7 STIMME    JSR  GETBYT  ;PARAMETER->X
95F7 CA                 DEX          ;AUS 1-3 MACH 0-2
95F8 E0 03              CPX  #3
95FA 90 02              BCC  STIMMM
95FC A2 00              LDX  #0      ;KORREKTUR
95FE 8E 48 C6 STIMMM    STX  ZWIS    ;STIMME
9601 60                 RTS
                 ;
                 ;
                 ;
```

```
               ;********************
               ;**                **
               ;**  BEFEHL: TUNE  **
               ;**                **
               ;********************
               ;
               ;
9602 20 F4 95 TUNE      JSR  STIMME  ;STIMME HOLEN
9605 20 FD AE           JSR  CHKCOM  ;TESTE AUF KOMMA
9608 20 EB B7           JSR  GETCOR  ;LAENGE(16BIT)+TON
960B 8A                 TXA          ;TON
960C C9 0D    TU7       CMP  #13
960E 90 04              BCC  TU8
9610 E9 0C              SBC  #12     ;KORREKTUR
9612 B0 F8              BCS  TU7     ;UNBEDINGT
               ;
9614 85 FE    TU8       STA  USE+1   ;TON
               ;
9616 20 F1 B7           JSR  CHKGET  ;OKTAVE -> X
9619 8A                 TXA          ;OKTAVE
961A 29 07              AND  #7      ;KORREKTUR
961C A6 FE              LDX  USE+1
961E E0 0C              CPX  #12
9620 D0 07              BNE  TU0
9622 A2 00              LDX  #0
9624 86 FE              STX  USE+1
9626 AA                 TAX
9627 E8                 INX          ;INC OKTAVE
9628 8A                 TXA          ;12.TON=H => 0.TON
9629 48       TU0       PHA
               ;
962A A2 00              LDX  #0
962C 20 79 00           JSR  CHRGOT
962F F0 03              BEQ  TU1     ;KEIN ZEICHEN MEHR
               ;
9631 20 F1 B7           JSR  CHKGET  ;HOLE VERSTIMMUNG
9634 86 FD    TU1       STX  USE     ;EVT. VERSTIMMUNG
9636 AC 48 C6           LDY  ZWIS    ;STIMME
9639 20 D1 96           JSR  TU6     ;WARTEN AUF STIMME AUS
               ;
963C A5 14              LDA  XK      ;TIMER
963E 78                 SEI
963F 99 31 C6           STA  TIMEL,Y
9642 A5 15              LDA  XK+1
9644 99 34 C6           STA  TIMEH,Y ;SETZEN
               ;
```

```
9647 A5 FE          LDA  USE+1    ;TON
9649 58             CLI
964A 0A             ASL  A          ;*2
964B A8             TAY
964C B9 D8 96       LDA  TONTAB,Y  ;HOLE BASISWERT
964F 85 AC          STA  ADL        ;FREQUENZ
9651 B9 D9 96       LDA  TONTAB+1,Y
9654 85 AD          STA  ADH
9656 68             PLA             ;OKTAVE
9657 49 07          EOR  #7
9659 A8             TAY
965A F0 07          BEQ  TU3        ;FERTIG
               ;
965C 46 AD     TU4  LSR  ADH
965E 66 AC          ROR  ADL        ;TONWERT/2
9660 88             DEY
9661 D0 F9          BNE  TU4
               ;
9663 AE 48 C6  TU3  LDX  ZWIS       ;POINTERBEST.
9666 BC F0 96       LDY  POITAB,X  ;STIMME*7
9669 A2 00          LDX  #0         ;X=0
966B A5 FD          LDA  USE        ;VERSTIMMUNG
966D 10 04          BPL  TU5        ;7.BIT=1 => MINUSVERST.
966F 49 7F          EOR  #$7F
9671 A2 FF          LDX  #$FF
9673 18        TU5  CLC
9674 65 AC          ADC  ADL      ;ZU FREQUENZ ADDIEREN
9676 99 00 D4       STA  S,Y      ;UND REALISIEREN
9679 8A             TXA
967A 65 AD          ADC  ADH
967C 99 01 D4       STA  S+1,Y    ;VERST.+TONWERT IN REG.
967F AE 48 C6       LDX  ZWIS     ;X: ;STIMME/Y: ;STIMME*7
               ;
9682 78             SEI
9683 BD 71 C6       LDA  WELLFO,X
9686 09 01          ORA  #1
9688 99 04 D4       STA  S+4,Y    ;TON ANSTELLEN
968B 9D 71 C6       STA  WELLFO,X
968E 58             CLI
968F 60             RTS
               ;
               ;
               ;
```

```
              ;***********************
              ;**                   **
              ;**  BEFEHL: FILTER   **
              ;**                   **
              ;***********************
              ;
              ;
9690 20 9E B7 FILTER    JSR  GETBYT  ;FILTERART -> X
9693 8A                 TXA          ;FILTERART
9694 0A                 ASL  A
9695 0A                 ASL  A
9696 0A                 ASL  A
9697 0A                 ASL  A       ;INS OBERE NIBBLE SCHIEBEN
9698 85 FE              STA  USE+1   ;MERKEN
969A AD 74 C6           LDA  VOLUM
969D 29 0F              AND  #$0F
969F 20 45 95           JSR  VO1     ;SETZEN
              ;
96A2 20 79 00           JSR  CHRGOT
96A5 F0 29              BEQ  FILO    ;KEIN ZEICHEN MEHR
              ;
96A7 20 FD AE           JSR  CHKCOM
96AA 20 EB B7           JSR  GETCOR  ;FREQUENZ UND RESONANZ
96AD A0 0B              LDY  #11
96AF A5 FD              LDA  USE
96B1 46 15    FIL1      LSR  XK+1
96B3 66 14              ROR  XK
96B5 6A                 ROR  A       ;FILTERFREQ.(NUR H-BYTE)
96B6 88                 DEY
96B7 D0 F8              BNE  FIL1
96B9 8D 16 D4           STA  S+22    ;REALISIEREN
              ;
96BC 8A                 TXA          ;RESONANZFREQ.
96BD 0A                 ASL  A
96BE 0A                 ASL  A
96BF 0A                 ASL  A
96C0 0A                 ASL  A
96C1 85 FD              STA  USE
96C3 AD 70 C6           LDA  FILON
96C6 29 0F              AND  #$0F
96C8 05 FD              ORA  USE
96CA 8D 17 D4           STA  S+23    ;SETZEN
96CD 8D 70 C6           STA  FILON
96D0 60       FILO      RTS
              ;
              ;
```

```
               ;AUF STIMME IN Y WARTEN:
               ;***********************
               ;
96D1 B9 71 C6 TU6      LDA  WELLFO,Y
96D4 4A                LSR  A
96D5 B0 FA             BCS  TU6      ;STIMME NOCH IN BETRIEB
96D7 60                RTS
               ;
               ;
               ;TABELLE DER TONWERTE EINER OKTAVE:
               ;**********************************
               ;
96D8 6A 83    TONTAB   .WOR 33642   ;H-6
96DA 3B 8B             .WOR 35643   ;C-7
96DC 82 93             .WOR 37762   ;C#-7
96DE 48 9C             .WOR 40008   ;D-7
96E0 93 A5             .WOR 42387   ;D#-7
96E2 6B AF             .WOR 44907   ;E-7
96E4 DA B9             .WOR 47578   ;F-7
96E6 E7 C4             .WOR 50407   ;F#-7
96E8 9C D0             .WOR 53404   ;G-7
96EA 04 DD             .WOR 56580   ;G#-7
96EC 28 EA             .WOR 59944   ;A-7
96EE 14 F8             .WOR 63508   ;A#-7
               ;
               ;
               ;POINTERTABELLE FUER 3 STIMMEN:
               ;******************************
               ;
96F0 00 07 0E POITAB   .BYT 0,7,14
               ;
               ;
               ;
               ;
               ;DEFINITION EINIGER SPEICHER:
               ;****************************
               ;
00A4          T1ADR    =    $A4
96F3 00       T1RVS    .BYT 0
96F4 00       T1Y2     .BYT 0
96F5 00       T1YR     .BYT 0
0061          T1CP     =    $61
96F6 00       T1CH     .BYT 0
96F7 00 00 00 T1BUF    .BYT 0,0,0,0,0,0,0,0
96FF 00 00 00 T1CHAD   .BYT 0,0,0,0,0,0,0,0
9707 00       T1LEN    .BYT 0
               ;
               ;
               ;
```

```
              ;*********************
              ;**                 **
              ;**  BEFEHL: TEXT   **
              ;**                 **
              ;*********************
              ;
              ;
9708 20 67 7C TEXT     JSR  STFLG    ;MODUSFLAGS HOLEN
970B 20 9E AD          JSR  FRMEVL   ;AUSDRUCK BEARBEITEN
970E 20 A3 B6          JSR  FRESTR
9711 8D 07 97          STA  T1LEN    ;STRINGLAENGE
9714 A5 22             LDA  $22
9716 A6 23             LDX  $23
9718 85 A4             STA  T1ADR
971A 86 A5             STX  T1ADR+1 ;STRINGADRESSE
              ;
971C 20 FD AE          JSR  CHKCOM   ;AUF KOMMA TESTEN
971F 20 FF 7F          JSR  TESCOR   ;KOORDINATEN HOLEN
9722 8D 42 C6          STA  X2L
9725 8E 43 C6          STX  X2H
9728 8C 40 C6          STY  Y2       ;SPEICHERN
              ;
972B A9 00             LDA  #0
972D 8D F3 96          STA  T1RVS    ;RVS-FLAG LOESCHEN
9730 20 F1 B7          JSR  CHKGET   ;MODUS->X
9733 8A                TXA
9734 29 01             AND  #1
9736 8D FF 96          STA  T1CHAD   ;UND MERKEN
              ;
9739 AD 40 C6          LDA  Y2       ;Y-KOORDINATE
973C C9 07             CMP  #7
973E B0 02             BCS  T1ILL    ;>=7
9740 A9 08             LDA  #8       ;KORREKTUR BEI ZU KLEIN
              ;
9742 A0 00    T1ILL    LDY  #0          ;POINTER AUF STRING
9744 B1 A4    T1LPO    LDA  (T1ADR),Y   ;ZEICHEN HOLEN
9746 8D F6 96          STA  T1CH        ;SICHERN
9749 8C F5 96          STY  T1YR        ;POINTER SICHERN
              ;
              ;AUF RVS ON/OFF TESTEN:
              ;
974C C9 12             CMP  #$12     ;RVS ON?
974E D0 08             BNE  T1L2     ;NEIN =>
9750 A9 80             LDA  #$80
9752 8D F3 96          STA  T1RVS    ;RVS-FLAG SETZEN
9755 4C 28 98          JMP  TNXTCH   ;NAECHSTES ZEICHEN
              ;
9758 C9 92    T1L2     CMP  #$92     ; RVS OFF?
975A D0 06             BNE  T1L3     ; NEIN =>
975C 0E F3 96          ASL  T1RVS    ;RVS-FLAG LOESCHEN
975F 4C 28 98          JMP  TNXTCH   ;NAECHSTES ZEICHEN
```

```
               ;
9762 C9 FF     T1L3     CMP  #$FF    ;PI?
9764 D0 05              BNE  T1L3A   ;NEIN
9766 A9 DE              LDA  #222    ;DURCH ANDEREN CODE ERSETZEN
9768 8D F6 96           STA  T1CH
               ;
               ;ASCII-CODE IN BILDSCHIRMCODE UMWANDLEN:
               ;
976B 29 20     T1L3A    AND  #$20
976D D0 13              BNE  T1L4
976F AD F6 96           LDA  T1CH
9772 29 BF              AND  #$BF
9774 8D F6 96           STA  T1CH
9777 29 80              AND  #$80
9779 4A                 LSR  A
977A 0D F6 96           ORA  T1CH
977D 29 7F              AND  #$7F
977F 8D F6 96           STA  T1CH
               ;
9782 AD F6 96  T1L4     LDA  T1CH
9785 0D F3 96           ORA  T1RVS   ;RVS-FLAG BERUECKSICHTIGEN
9788 A0 1A              LDY  #$1A    ;$D0 / 3
978A AE FF 96           LDX  T1CHAD  ;MODUS
978D E0 00              CPX  #0
978F F0 02              BEQ  T1L4A
9791 A0 1B              LDY  #$1B
9793 85 61     T1L4A    STA  T1CP    ;ADRESSE LOW-BYTE
9795 84 62              STY  T1CP+1  ;ADRESSE HIGH-BYTE
9797 06 61              ASL  T1CP
9799 26 62              ROL  T1CP+1
979B 06 61              ASL  T1CP
979D 26 62              ROL  T1CP+1
979F 06 61              ASL  T1CP
97A1 26 62              ROL  T1CP+1  ;ADRESSE*8
               ;
97A3 78                 SEI
97A4 A5 01              LDA  1
97A6 48                 PHA
97A7 29 FB              AND  #$FB
97A9 85 01              STA  1       ;ROM AUS
               ;
97AB A0 07              LDY  #7
97AD B1 61     T1L5     LDA  (T1CP),Y
97AF 99 F7 96           STA  T1BUF,Y ;ZEICHENMATRIX->BUFFER
97B2 88                 DEY
97B3 10 F8              BPL  T1L5
               ;
97B5 68                 PLA
97B6 85 01              STA  1       ;ROM AN
97B8 58                 CLI
               ;
```

```
               ;ZEICHENMATRIX NACH GRAPHIK:
               ;
97B9 A0 08               LDY  #8       ;SPALTENZAEHLER
97BB A2 07     T1LP1     LDX  #7       ;ZEILENZAEHLER
97BD BD F7 96  T1LP2     LDA  T1BUF,X  ;MATRIX AUS BUFFER
97C0 0A                  ASL  A        ;BIT HERAUSSCHIEBEN
97C1 9D F7 96            STA  T1BUF,X  ;REST WIEDER ZURUECK
97C4 08                  PHP           ;BIT MERKEN
97C5 8E F4 96            STX  T1Y2     ;ZEILENZAEHLER MERKEN
97C8 A9 07               LDA  #7
97CA 38                  SEC
97CB ED F4 96            SBC  T1Y2
97CE 8D F4 96            STA  T1Y2     ;7-ZEILE
97D1 AD 40 C6            LDA  Y2
97D4 38                  SEC
97D5 ED F4 96            SBC  T1Y2
97D8 8D F4 96            STA  T1Y2     ;Y-(7-ZEILE) = Y+ZEILE-7
97DB 28                  PLP           ;BIT
97DC 90 14               BCC  T1L6     ;NICHT GESETZT->KEIN PUNKT
               ;
97DE 98                  TYA
97DF 48                  PHA
97E0 8A                  TXA
97E1 48                  PHA           ;ZEILEN-/SPALTENZAEHLER MERKEN
97E2 AE 43 C6            LDX  X2H
97E5 AD 42 C6            LDA  X2L
97E8 AC F4 96            LDY  T1Y2     ;KOORDINATEN HOLEN
97EB 20 47 80            JSR  PLOT.    ;PUNKT SETZEN
97EE 68                  PLA
97EF AA                  TAX
97F0 68                  PLA
97F1 A8                  TAY           ;ZAEHLER WIEDERHOLEN
97F2 CA        T1L6      DEX
97F3 10 C8               BPL  T1LP2    ;NAECHSTE ZEILE
97F5 EE 42 C6            INC  X2L
97F8 D0 03               BNE  T1L7
97FA EE 43 C6            INC  X2H      ;X-KOORDINATE INCREMIEREN
               ;
97FD A5 02     T1L7      LDA  GFLAG
97FF 4A                  LSR  A
9800 AE 42 C6            LDX  X2L
9803 AD 43 C6            LDA  X2H
9806 B0 14               BCS  T1L9     ;LGR
               ;
9808 24 02               BIT  GFLAG
980A 30 08               BMI  T1L10    ;MC
               ;
```

```
              ;AUF BILDSCHIRMRAENDER TESTEN:
              ;
980C E0 40              CPX  #<320
980E E9 01              SBC  #>320
9810 90 10              BCC  T1L8
9812 B0 20              BCS  T1EXIT
              ;
9814 E0 A0    T1L10     CPX  #<160
9816 E9 00              SBC  #>160
9818 90 08              BCC  T1L8
981A B0 18              BCS  T1EXIT
              ;
981C D0 16    T1L9      BNE  T1EXIT
981E E0 50              CPX  #<80
9820 B0 12              BCS  T1EXIT
              ;
              ;
9822 88       T1L8      DEY
9823 F0 03              BEQ  TNXTCH  ;NAECHSTE SPALTE
9825 4C BB 97           JMP  T1LP1
              ;
              ;NAECHSTES ZEICHEN:
              ;
9828 AC F5 96 TNXTCH    LDY  T1YR
982B C8                 INY
982C CC 07 97           CPY  T1LEN
982F F0 03              BEQ  T1EXIT  ;FERTIG
9831 4C 44 97           JMP  T1LP0
              ;
9834 4C C4 7F T1EXIT    JMP  PLTVAR  ;TEST?
              ;
              ;
              ;
              ;
              ;
              ;**********************
              ;**                  **
              ;**  BEFEHL: CIRCLE  **
              ;**                  **
              ;**********************
              ;
              ;
C65E          CMODE     =    $1+SAD
C668          CXC       =    $B+SAD
C66A          CYC       =    $D+SAD
C665          CXG       =    $8+SAD
C660          CYG       =    $3+SAD
C661          CW1       =    $4+SAD
C663          CW2       =    $6+SAD
              ;
              ;
```

```
9837 20 67 7C CIRC     JSR  STFLG   ;HOLE MODI
983A 20 79 00          JSR  CHRGOT
983D C9 2C             CMP  #","    ;KOMMA?
983F 08       CIRC0    PHP
9840 A9 02             LDA  #2      ;2 FUER GENAUIGKEIT ALS VORGABE
9842 85 0F             STA  HKFLG1  ;FLAG LOESCHEN
9844 28                PLP          ;KOMMA?
9845 F0 06             BEQ  CIRC1   ;JA->KEIN SCHRITTPARAMETER
9847 20 9E B7          JSR  GETBYT  ;SCHRITTWEITE->X
984A 8A                TXA          ;SCHRITTWEITE PRO BERECHNUNG IN GRAD
984B F0 F2             BEQ  CIRC0   ;=0 DANN 2 (NORMAL)
              ;
              ;MITTELPUNKTSKOORDINATEN HOLEN:
              ;
984D 48       CIRC1    PHA          ;SCHRITTWEITE MERKEN
984E 20 FD AE          JSR  CHKCOM  ;AUF KOMMA TESTEN
9851 20 FF 7F          JSR  TESCOR  ;MITTELPUNKTSKOORDINATEN
9854 8D 42 C6          STA  X2L
9857 8D 68 C6          STA  CXC
985A 8E 43 C6          STX  X2H
985D 8E 69 C6          STX  CXC+1
9860 8C 40 C6          STY  Y2
9863 8C 6A C6          STY  CYC     ;MITTELPUNKTSKOORDINATEN
              ;
              ;X/Y-RADIUS HOLEN:
              ;
9866 20 FD AE          JSR  CHKCOM  ;AUF KOMMA TESTEN
9869 A9 00             LDA  #0
986B 8D 67 C6          STA  $A+SAD
986E 8D 5D C6          STA  $0+SAD
9871 20 FF 7F          JSR  TESCOR  ;X/Y-RADIUS HOLEN
9874 8D 65 C6          STA  CXG
9877 8E 66 C6          STX  CXG+1
987A 8C 60 C6          STY  CYG     ;X/Y-RADIUS
              ;
              ;START- UND ENDWINKEL HOLEN:
              ;
987D A9 00             LDA  #0
987F 8D 61 C6          STA  CW1
9882 8D 62 C6          STA  CW1+1   ;FEHLPARAMETER STARTWINKEL
9885 A9 68             LDA  #<360
9887 A0 01             LDY  #>360
9889 8D 63 C6          STA  CW2
988C 8C 64 C6          STY  CW2+1   ;FEHLPARAMETER ENDWINKEL
              ;
988F 20 79 00          JSR  CHRGOT  ;LETZTES ZEICHEN HOLEN
9892 F0 12             BEQ  CL2     ;KEINE PARAMETER MEHR
```

```
              ;
9894 20 91 9A           JSR  GETGRA
9897 8C 61 C6           STY  CW1
989A 8D 62 C6           STA  CW1+1    ;STARTWINKEL
989D 20 91 9A           JSR  GETGRA
98A0 8C 63 C6           STY  CW2
98A3 8D 64 C6           STA  CW2+1    ;ENDWINKEL
              ;
              ;WINKELWERTE UEBERPRUEFEN:
              ;
98A6 AC 61 C6 CL2       LDY  CW1
98A9 AD 62 C6           LDA  CW1+1
98AC C0 68              CPY  #<360
98AE E9 01              SBC  #>360
98B0 90 03              BCC  CL3
98B2 4C 48 B2 CL4       JMP  QERR
              ;
98B5 AC 63 C6 CL3       LDY  CW2
98B8 AD 64 C6           LDA  CW2+1
98BB CC 61 C6           CPY  CW1
98BE ED 62 C6           SBC  CW1+1
98C1 B0 05              BCS  CLQW     ;ENDWINKEL >= STARTWINKEL
98C3 A9 FF              LDA  #$FF
98C5 8D 5D C6           STA  $0+SAD   ;FLAG FUER ENDWINKEL < STARTWINKEL
98C8 C0 69    CLQW      CPY  #<361
98CA AD 64 C6           LDA  CW2+1
98CD E9 01              SBC  #>361
98CF B0 E1              BCS  CL4
              ;
              ;HAUPTVERTEILER:
              ;
98D1 68                 PLA           ;SCHRITTWEITE
98D2 85 AE              STA  SHP
98D4 20 D4 7C           JSR  TAZ      ;ADRESSE HOLEN
98D7 20 E0 98           JSR  CNEXT5   ;ELLIPSE ZEICHNEN
98DA 20 F7 7C           JSR  TZA      ;ADRESSE RETTEN
98DD 4C C4 7F           JMP  PLTVAR   ;TEST?
              ;
              ;
              ;
```

```
              ;ELLIPSE ZEICHNEN:
              ;
              ;NACH DER FORMEL:
              ;X = XR*SIN(W) + MX
              ;Y = YR*SIN(W) + MY
              ;
              ;WOBEI:
              ; X, Y: KOORDINATEN DES KREISPUNKTES
              ;XR,YR: X,Y-RADIUS
              ;MX,MY: MITTELPUNKTKOORDINATEN
              ;    W: AKTUELLER KREISWINKEL
              ;
              ;
              ;BERECHNE SIN(W1), COS(W1):
              ;NACH COS(W1) = SIN(W1+90)
              ;
98E0 AD 61 C6 CNEXT5  LDA  CW1
98E3 18               CLC
98E4 69 5A            ADC  #90
98E6 AA               TAX
98E7 AD 62 C6         LDA  CW1+1
98EA 69 00            ADC  #0       ;W1+90 BERECHNEN
98EC 4A               LSR  A
98ED 8A               TXA
98EE 6A               ROR  A        ;/2 FUER TABELLE
              ;
              ;WERT AUF 0-89 BRINGEN:
              ;
98EF 38               SEC
98F0 AA       CSUB1   TAX
98F1 E9 5A            SBC  #90      ;>=90?
98F3 B0 FB            BCS  CSUB1    ;JA => 90 ABZIEHEN
              ;
              ;
98F5 AD 62 C6         LDA  CW1+1
98F8 4A               LSR  A
98F9 AD 61 C6         LDA  CW1
98FC 6A               ROR  A
98FD 38               SEC
98FE A8       CSUB    TAY
98FF E9 5A            SBC  #90      ;>=90?
9901 B0 FB            BCS  CSUB     ;JA => 90 ABZIEHEN
              ;
              ;WERTE AUS TABELLE HOLEN:
              ;
9903 B9 AB 9A         LDA  CSTAB,Y ;SIN(W1) (0<=SIN(W1)<=1)
9906 8D 40 C6         STA  Y2       ;NACH Y2
              ;
9909 BD AB 9A         LDA  CSTAB,X ;SIN(W1+90)
990C 8D 56 C6         STA  C1       ;NACH 8-BIT-FAKTOR
              ;
```

```
990F AD 60 C6          LDA  CYG      ;Y-RADIUS
9912 A0 00             LDY  #0       ;HIGH-BYTE
9914 8D 58 C6          STA  C3       ;NACH 16-BIT-FAKTOR
9917 8C 59 C6          STY  C4
               ;
               ;
               ;MULTIPLIKATION:
               ;
               ;8-BIT-WERT * 16-BIT-WERT -> 24-BIT-WERT
               ;    C1     *   C3/C4     -> C2/X2L/X2H
               ;
               ;DABEI GILT: 0 <= C1 <= 1
               ;UND DAMIT HAT DAS ERGEBNIS
               ;8 NACHKOMMABITS (IN C2),
               ;DIE VERNACHLAESSIGT WERDEN KOENNEN
               ;
               ;
991A 18                CLC           ;FLAG FUER ERSTE MULTIPLIKATION
991B 08        CMUL    PHP           ;FLAG MERKEN
991C A9 00             LDA  #0
991E 8D 42 C6          STA  X2L
9921 8D 43 C6          STA  X2H
9924 8D 57 C6          STA  C2       ;ERGEBNISSPEICHER=0
               ;
9927 A2 08             LDX  #8       ;8 DURCHLAEUFE (FUER JEDES BIT)
9929 0E 57 C6  CMUL17  ASL  C2
992C 2E 42 C6          ROL  X2L      ;ERGEBNISSPEICHER EINE DUAL-
992F 2E 43 C6          ROL  X2H      ;STELLE WEITERSCHIEBEN
               ;
9932 0E 56 C6          ASL  C1       ;EIN BIT AUS FAKTOR SCHIEBEN
9935 90 1B             BCC  CMUL16   ;FALLS 0=>KEINE ADDITION
9937 18                CLC           ;16-BIT-FAKTOR ZU ERGEBNIS ADDIEREN:
9938 AD 57 C6          LDA  C2
993B 6D 58 C6          ADC  C3
993E 8D 57 C6          STA  C2       ;LOW-BYTE
9941 AD 42 C6          LDA  X2L
9944 6D 59 C6          ADC  C4
9947 8D 42 C6          STA  X2L      ;MID-BYTE
994A AD 43 C6          LDA  X2H
994D 69 00             ADC  #0
994F 8D 43 C6          STA  X2H      ;HIGH-BYTE
               ;
9952 CA        CMUL16  DEX           ;BIT-ZAEHLER DECREMIEREN
9953 D0 D4             BNE  CMUL17   ;NAECHSTES BIT
```

```
              ;MULTIPLIKATION ZUENDE
              ;---------------------------
              ;
              ;ZWEITE MULTIPLIKATION VORBEREITEN:
              ;
9955 28               PLP            ;FLAG
9956 B0 23            BCS   CMUL1    ;BEREITS 2. DURCHLAUF
              ;
9958 AD 40 C6         LDA   Y2
995B 8D 56 C6         STA   C1       ;SIN(W1) NACH 8-BIT-FAKTOR
              ;
995E AD 42 C6         LDA   X2L      ;ERSTES BYTE VOR DEM KOMMA
9961 8D 40 C6         STA   Y2       ;NACH Y2
              ;
9964 AD 65 C6         LDA   CXG
9967 AC 66 C6         LDY   CXG+1
996A 8D 58 C6         STA   C3
996D 8C 59 C6         STY   C4       ;X-RADIUS NACH 16-BIT-FAKTOR
9970 AC 43 C6         LDY   X2H      ;ZWEITES BYTE VOR DEM KOMMA>0?
9973 D0 03            BNE   CL101    ;JA->FERTIG, DA WERT ZU HOCH
              ;
9975 38               SEC            ;FLAG FUER 2. DURCHLAUF
9976 B0 A3            BCS   CMUL     ;UNBEDINGT Z. 2. MULT.
              ;
9978 4C 89 9A CL101   JMP   CEXIT
              ;
              ;MOMENTANE VARIABLENBELEGUNG:
              ;
              ;X2L: XR*SIN(W1) //LOW-BYTE
              ;X2H: XR*SIN(W1) //HIGH-BYTE
              ;Y2 : YR*COS(W1)
              ;
              ;GRADZAHL TESTEN:
              ;(MITTELPUNKTKOORDINATEN BEARBEITEN)
              ;
997B AC 61 C6 CMUL1   LDY   CW1      ;W1=LAUFENDE GRADZAHL
997E AD 62 C6         LDA   CW1+1
9981 C0 5A            CPY   #90
9983 E9 00            SBC   #0
9985 90 28            BCC   CNEGY    ;W1 < 90 => OBERE KREISHAELFTE
9987 AD 62 C6         LDA   CW1+1
998A C0 0E            CPY   #<270
998C E9 01            SBC   #>270
998E B0 1F            BCS   CNEGY    ;W1 >= 270 => OBERE KREISHAELFTE
              ;
```

```
                ;Y-MITTELPUNKTKOORDINATE ZUADDIEREN:
                ;(FUER UNTERE KREISHAELFTE)
                ;
9990 AD 6A C6            LDA  CYC
9993 18                  CLC
9994 6D 40 C6            ADC  Y2
9997 8D 40 C6            STA  Y2       ;YC+Y2->Y2
999A B0 DC               BCS  CL101    ;ZU HOCH
999C 48                  PHA
                ;
999D A5 02               LDA  GFLAG
999F 4A                  LSR  A
99A0 68                  PLA
99A1 90 06               BCC  CI4      ;->HGR/MC
99A3 C9 32               CMP  #50      ;LGR-TEST
99A5 90 14               BCC  CRDYY    ;Y<50 -> OK.
99A7 B0 CF               BCS  CL101    ;Y>=50 -> ENDE
                ;
99A9 C9 C8      CI4      CMP  #200     ;HGR/MC
99AB 90 0E               BCC  CRDYY    ;Y<200 -> OK.
99AD B0 C9      CL102    BCS  CL101    ;Y>=200 -> ENDE
                ;
                ;KOORD. V. Y-MITTELP.KOORD. ABZIEHEN:
                ;(FUER UNTERE KREISHAELFTE)
                ;
99AF 38         CNEGY    SEC
99B0 AD 6A C6            LDA  CYC
99B3 ED 40 C6            SBC  Y2
99B6 8D 40 C6            STA  Y2       ;YC-Y2->Y2
99B9 90 BD      CL103    BCC  CL101    ;Y<0 -> ENDE
                ;
                ;GRADZAHL TESTEN:
                ;
99BB AD 62 C6   CRDYY    LDA  CW1+1    ;LAUFENDE GRADZAHL
99BE D0 3F               BNE  CNEGX    ;>180 -> LINKE KREISHAELFTE
99C0 AD 61 C6            LDA  CW1
99C3 C9 B4               CMP  #180
99C5 B0 38               BCS  CNEGX    ;>180 -> LINKE KREISHAELFTE
                ;
                ;X-MITTELP.KOORD. ZUADDIEREN:
                ;(FUER RECHTE KREISHAELFTE)
                ;
99C7 18                  CLC
99C8 AD 42 C6            LDA  X2L
99CB 6D 68 C6            ADC  CXC
99CE 8D 42 C6            STA  X2L
99D1 A8                  TAY
99D2 AD 43 C6            LDA  X2H
99D5 6D 69 C6            ADC  CXC+1
99D8 8D 43 C6            STA  X2H      ;XC+X2->X2
99DB B0 D0               BCS  CL102    ;ZU HOCH
```

```
99DD 48              PHA
               ;
99DE A5 02           LDA   GFLAG    ;GRAPHIKFLAG
99E0 10 14           BPL   CI6      ;->HGR
99E2 4A              LSR   A
99E3 68              PLA
99E4 B0 08           BCS   CI5      ;->LGR
99E6 C0 A0           CPY   #<160    ;MC
99E8 E9 00           SBC   #>160
99EA 90 28           BCC   CRDYX    ;X< 160 -> OK.
99EC B0 BF           BCS   CL102    ;X>=160 -> ENDE
               ;
99EE C0 50     CI5   CPY   #<80
99F0 E9 00           SBC   #>80
99F2 90 20           BCC   CRDYX    ;X< 80 -> OK.
99F4 B0 B7           BCS   CL102    ;X>=80 -> ENDE
               ;
99F6 68        CI6   PLA
99F7 C0 40           CPY   #<320
99F9 E9 01           SBC   #>320
99FB 90 17           BCC   CRDYX    ;X< 320 -> OK.
99FD B0 AE           BCS   CL102    ;X>=320 -> ENDE
               ;
               ;X-KOORD. V. MITTELP.KOORD. ABZIEHEN:
               ;(FUER LINKE KREISHAELFTE)
               ;
99FF AD 68 C6 CNEGX  LDA   CXC
9A02 38              SEC
9A03 ED 42 C6        SBC   X2L
9A06 8D 42 C6        STA   X2L
9A09 AD 69 C6        LDA   CXC+1
9A0C ED 43 C6        SBC   X2H
9A0F 8D 43 C6        STA   X2H      ;XC-X2->X2
9A12 90 A5           BCC   CL103    ;X<0 -> ENDE
               ;
               ;
               ;ZEICHNEN:
               ;
9A14 AD 42 C6 CRDYX  LDA   X2L
9A17 AE 43 C6        LDX   X2H
9A1A AC 40 C6        LDY   Y2       ;KOORDINATEN -> REGISTER
9A1D 2C 67 C6        BIT   $A+SAD   ;ERSTFLAG
9A20 10 09           BPL   CMOVE    ;ERSTER ERRECHNETER PUNKT
9A22 20 5E 80        JSR   LINE.    ;LINIE VOM LETZTEN PUNKT
9A25 38              SEC
9A26 B0 0B           BCS   CCONT    ;UNBEDINGT WEITER
               ;
9A28 4C E0 98 CNEXT  JMP   CNEXT5   ;NAECHSTEN PUNKT BERECHNEN
               ;
```

```
                ;GRAPHIKCURSOR AUF ERSTEN PUNKT:
                ;
9A2B 20 B0 7F CMOVE    JSR  SETCUR  ;SETZE CURSOR
9A2E A9 FF             LDA  #$FF
9A30 8D 67 C6          STA  $A+SAD  ;ERST-FLAG LOESCHEN
                ;
                ;
                ;WINKEL ERHOEHEN:
                ;
9A33 18       CCONT    CLC
9A34 AD 61 C6          LDA  CW1
9A37 65 AE             ADC  SHP     ;WINKELABSTAND
9A39 8D 61 C6          STA  CW1     ;ZU STARTWINKEL ADDIEREN
9A3C A8                TAY
9A3D AD 62 C6          LDA  CW1+1
9A40 69 00             ADC  #0
9A42 8D 62 C6          STA  CW1+1   ;HIGH-BYTE
                ;
9A45 2C 5D C6          BIT  $0+SAD  ;FLAG
9A48 10 1C             BPL  CLQ1
                ;
                ;ENDWINKEL WAR KLEINER ALS STARTWINKEL:
                ;
9A4A C0 69             CPY  #<361
9A4C E9 01             SBC  #>361   ;LFD. WINKEL > 360?
9A4E 90 D8             BCC  CNEXT   ;NEIN->WEITER
9A50 A9 00             LDA  #0      ;JA->360 ABZIEHEN
9A52 8D 5D C6          STA  $0+SAD  ;FLAG LOESCHEN
9A55 AD 61 C6          LDA  CW1     ;C=1!
9A58 E9 68             SBC  #<360
9A5A 8D 61 C6          STA  CW1
9A5D A8                TAY
9A5E AD 62 C6          LDA  CW1+1
9A61 E9 01             SBC  #>360
9A63 8D 62 C6          STA  CW1+1   ;W1-360->W1
                ;
9A66 CC 63 C6 CLQ1     CPY  CW2
9A69 08                PHP          ;ERGEBNIS FUER LOW-BYTES MERKEN
9A6A ED 64 C6          SBC  CW2+1   ;-ENDWINKEL
9A6D F0 1B             BEQ  CNEXTQ  ;HIGH-BYTES GLEICH
9A6F 68                PLA
9A70 90 B6             BCC  CNEXT   ;NOCH NICHT FERTIG
                ;
9A72 A5 0F    CPSEX    LDA  HKFLG1  ;FAST FERTIG:
9A74 49 FF             EOR  #$FF
9A76 85 0F             STA  HKFLG1  ;FLAG FUER NOCH EINMAL
9A78 10 0F             BPL  CEXIT   ;FERTIG
                ;
```

```
              ;DAS LETZTE MAL BIS ENDGRAD ZEICHNEN:
              ;
9A7A AD 63 C6          LDA  CW2
9A7D 8D 61 C6          STA  CW1
9A80 AD 64 C6          LDA  CW2+1
9A83 8D 62 C6          STA  CW1+1    ;BIS ENDGRAD ZEICHNEN
9A86 4C E0 98          JMP  CNEXT5
              ;FERTIG:
9A89 60       CEXIT    RTS           ;BEI GROESSER FERTIG
              ;
              ;FERTIG?
              ;
9A8A 28       CNEXTQ   PLP           ;LOW-BYTE PRUEFEN
9A8B F0 9B             BEQ  CNEXT
9A8D 90 99             BCC  CNEXT    ;NAECHSTER PUNKT
9A8F B0 E1             BCS  CPSEX    ;->FAST FERTIG
              ;
              ;
              ;GRADZAHL HOLEN:
              ;
9A91 20 FD AE GETGRA   JSR  CHKCOM   ;HOLE GRADZAHL
9A94 20 8A AD          JSR  $AD8A
9A97 20 F7 B7          JSR  $B7F7
9A9A A4 14             LDY  $14
9A9C A5 15             LDA  $15
9A9E D0 0A             BNE  GETGR1
9AA0 C0 FF             CPY  #255
9AA2 F0 04             BEQ  GETGR2
9AA4 C0 FE             CPY  #254
9AA6 D0 02             BNE  GETGR1
9AA8 A0 FD    GETGR2   LDY  #253
9AAA 60       GETGR1   RTS
              ;
              ;
              ;
              ;TABELLE ALLER 8-BIT-SINUSWERTE
              ;GEORDNET NACH GRAD VON 0-89
              ;IN 2-GRAD-ABSTAENDEN:
              ;********************************
              ;
              ;DIE WERTE SIND ALS WERTE VON
              ;0 BIS 1 (EXCL.) ZU VERSTEHEN
              ;(ANGEGEBEN WERDEN ALSO STETS DIE
              ; ERSTEN 8 BIT HINTER DEM KOMMA;
              ; S. AUCH MULTIPLIKATION)
              ;
              ;
```

```
9AAB 00 09 12 CSTAB     .BYT $00,$09,$12,$1B,$23,$2C,$35,$3E,$46,$4F
9AB5 57 60 68           .BYT $57,$60,$68,$70,$78,$7F,$87,$8F,$96,$9D
9ABF A4 AB B1           .BYT $A4,$AB,$B1,$B7,$BE,$C3,$C9,$CE,$D3,$D8
9AC9 DD E1 E5           .BYT $DD,$E1,$E5,$E9,$EC,$F0,$F3,$F5,$F7,$F9
9AD3 FB FD FE           .BYT $FB,$FD,$FE,$FE,$FF,$FF,$FF,$FE,$FE,$FD
9ADD FB F9 F7           .BYT $FB,$F9,$F7,$F5,$F3,$F0,$EC,$E9,$E5,$E1
9AE7 DD D8 D3           .BYT $DD,$D8,$D3,$CE,$C9,$C3,$BE,$B7,$B1,$AB
9AF1 A4 9D 96           .BYT $A4,$9D,$96,$8F,$87,$80,$78,$70,$68,$60
9AFB 57 4F 46           .BYT $57,$4F,$46,$3E,$35,$2C,$23,$1B,$12,$09
                ;
                ;
                ;
                ;
                ;
                ;*********************
                ;**                 **
                ;**  BEFEHL: MERGE  **
                ;**                 **
                ;*********************
                ;
                ;
9B05 20 D4 E1 MERGE     JSR  GETPAR   ;FILEPARAMETER HOLEN
9B08 A5 2D              LDA  VARSTA   ;START DER VARIABLEN
9B0A 38                 SEC
9B0B E9 02              SBC  #2       ;VARIABLENSTART-2 = STARTADRESSE
9B0D AA                 TAX
9B0E A5 2E              LDA  VARSTA+1
9B10 E9 00              SBC  #0       ;HIGH-BYTE
9B12 A8                 TAY
                ;
9B13 A9 00              LDA  #0       ;LOAD-FLAG
9B15 85 0A              STA  $0A      ;LOAD-FLAG DES INTERPRETERS
9B17 85 B9              STA  SECADR   ;SECUNDAERADRESSE=0
9B19 4C 75 E1           JMP  $E175    ;LADEN/ZEIGER SETZEN/ZEILEN BINDEN
                ;
                ;
                ;
                ;**********************
                ;**                  **
                ;**  BEFEHL: DTASET  **
                ;**                  **
                ;**********************
                ;
                ;
9B1C 20 8A AD DATAST    JSR  FRMNUM   ;NUMERISCHER WERT->FAC
9B1F 20 F7 B7           JSR  FACINT   ;NACH INTEGER ($14/$15)
                ;
9B22 A5 7A              LDA  PRGZG
9B24 48                 PHA           ;PROGRAMMZEIGER RETTEN
9B25 A5 7B              LDA  PRGZG+1
9B27 48                 PHA
```

```
                ;
9B28 20 A3 A8           JSR  GOTO    ;SUCHT ZEILE (GOTO-BEFEHL)
9B2B A5 7A              LDA  PRGZG
9B2D A4 7B              LDY  PRGZG+1
9B2F 20 27 A8           JSR  $A827   ;DATA-ZEIGER SETZEN
                ;
9B32 68                 PLA
9B33 85 7B              STA  PRGZG+1
9B35 68                 PLA
9B36 85 7A              STA  PRGZG   ;ALTER PROGRAMMZEIGER
9B38 60                 RTS
                ;
                ;
                ;
                ;*********************
                ;**                 **
                ;**  BEFEHL: RENUM  **
                ;**                 **
                ;*********************
                ;
                ;
9B39 20 EB B7 RENUM     JSR  GETCOR  ;STARTNUMMER/DIFFERENZ HOLEN
9B3C 8A                 TXA          ;DIFFERENZ
9B3D D0 03              BNE  RENUM.  ;OK.
9B3F 4C 48 B2 TSCHAU    JMP  QERR    ;=0-> ILLEGAL QUANTITY
                ;
9B42 86 97    RENUM.    STX  FLG     ;DIFFERENZ (MAX. 255)
9B44 A5 14              LDA  XK
9B46 48                 PHA
9B47 A5 15              LDA  XK+1
9B49 48                 PHA          ;STARTNUMMER IN XK RETTEN
                ;
                ;PASS 1:
                ;TESTEN, OB ZEILENNUMMERN ZU HOCH WERDEN
                ;
9B4A A9 80              LDA  #$80
9B4C 85 96              STA  FLG2    ;1.PASSFLAG
9B4E 20 BF 9C           JSR  PASSES  ;PASSDURCHGANG 1
9B51 B0 EC              BCS  TSCHAU  ;ZEILENNUMMER ZU HOCH!
                ;
                ;
9B53 A9 C1              LDA  #$C1
9B55 A0 FE              LDY  #$FE    ;RTI-ADRESSE IM ROM
9B57 8D 18 03           STA  IRQ+4
9B5A 8C 19 03           STY  IRQ+5   ;NMI-VEKTOR AUF RTI
                ;
```

```
                ;2.PASS:
                ;ALTE/NEUE ZEILENNUMMERN
                ;REFERENZTABELLE ERSTELLEN
                ;UND ZEILEN UMNUMERIEREN
                ;
9B5D A9 00              LDA  #0
9B5F 85 96              STA  FLG2     ;2.PASSFLAG
9B61 68                 PLA
9B62 85 15              STA  XK+1
9B64 68                 PLA
9B65 85 14              STA  XK       ;STARTNUMMER WIEDERHOLEN
9B67 20 BF 9C           JSR  PASSES   ;2. PASS DURCHFUEHREN
                ;
                ;3.PASS:
                ;GOTO/GOSUB ETC. AENDERN
                ;
9B6A 20 8E A6           JSR  $A68E    ;PROGRAMMZEIGER AUF BASIC-START
9B6D 4C 9E 9C           JMP  ENDZ1    ;STARTEINSPRUNG
                ;
                ;
                ;
                ;PASS 3 - SCHLEIFE:
                ;AENDERUNG VON GOTO/GOSUB ETC.
                ;
9B70 A0 03      REN5    LDY  #3
9B72 B1 7A              LDA  (PRGZG),Y  ;ZEILENNR.-LOW
9B74 48                 PHA             ;AUS PROGRAMM HOLEN
9B75 C8                 INY             ;UND
9B76 B1 7A              LDA  (PRGZG),Y  ;(HIGH)
9B78 48                 PHA             ;MERKEN
9B79 C8                 INY
9B7A 20 FB A8           JSR  $A8FB    ;Y ZU PROGRAMMZEIGER ADDIEREN
                ;
9B7D A2 03              LDX  #3       ;PUFFERPOINTER=3
9B7F 86 0F              STX  $0F      ;HOCHKOMMAFLAG LOESCHEN
9B81 86 08              STX  $08      ;REM-FLAG LOESCHEN
                ;
                ;ZEILENSCHLEIFE:
                ;****************
                ;
9B83 A0 FF      REN6    LDY  #$FF     ;Y ALS ZEIGER AUF ZEICHEN (OFFSET)
                ;
```

```
              ;ZEICHENSCHLEIFE:
              ;*****************
              ;
9B85 C8       REN7      INY              ;NAECHSTES ZEICHEN IN PRG
9B86 E8                 INX              ;NAECHSTES ZEICHEN IN PUFFER
9B87 B1 7A              LDA  (PRGZG),Y  ;LAUFENDES BYTE
9B89 9D FC 01           STA  $1FC,X      ;IN PUFFER COPIEREN
9B8C F0 0D              BEQ  ENDZ2       ;=0 => ZEILENENDE
              ;
9B8E C9 22              CMP  #$22      ;ANFUEHRUNGSZEICHEN?
9B90 D0 0C              BNE  REN8.0    ;NEIN
              ;
9B92 A5 0F              LDA  $0F       ;HOCHKOMMAFLAG
9B94 49 FF              EOR  #$FF      ;UMDREHEN
9B96 85 0F              STA  $0F
9B98 4C 85 9B           JMP  REN7      ;NAECHSTES BYTE
              ;
9B9B 4C 62 9C ENDZ2     JMP  ENDZEI    ;ZEILENENDE BEARBEITEN
              ;
9B9E 24 0F    REN8.0    BIT  $0F       ;HOCHKOMMAFLAG
9BA0 30 E3              BMI  REN7      ;BEREITS GESETZT!
9BA2 24 08              BIT  $08       ;REM-FLAG
9BA4 30 DF              BMI  REN7      ;BEREITS GESETZT!
              ;
9BA6 C9 8F              CMP  #$8F      ;REM?
9BA8 D0 04              BNE  REN8      ;NEIN
              ;
9BAA 85 08              STA  $08       ;REM-FLAG SETZEN
9BAC F0 D7              BEQ  REN7      ;UNBEDINGT
              ;
              ;AUF BEFEHL MIT ZEILENNUMMER TESTEN:
              ;
9BAE 86 97    REN8      STX  FLG        ;PUFFERPOINTER RETTEN
9BB0 A2 04              LDX  #4
9BB2 DD BA 9C REN8.2    CMP  RENDAT,X ;ZEILENNUMMERNBEFEHL?
9BB5 F0 08              BEQ  REN8.1     ;JA
9BB7 CA                 DEX             ;GANZE TABELLE TESTEN
9BB8 10 F8              BPL  REN8.2
              ;
9BBA A6 97              LDX  FLG       ;PUFFERPOINTER WIEDERHOLEN
9BBC 4C 85 9B           JMP  REN7      ;NAECHSTES BYTE
              ;
              ;
```

```
                ;BEFEHL MIT FOLGENDER
                ;ZEILENNUMMER BEARBEITEN:
                ;************************
                ;
9BBF 20 FB A8 REN8.1    JSR  $A8FB      ;Y-REG. ZU PRGZG ADDIEREN
9BC2 E6 7A    REN8.4    INC  PRGZG
9BC4 D0 02              BNE  REN8.5
9BC6 E6 7B              INC  PRGZG+1    ;PROGRAMMZEIGER+1
9BC8 A0 00    REN8.5    LDY  #0
9BCA B1 7A              LDA  (PRGZG),Y ;HOLE ZEICHEN AUS PROGRAMM
9BCC C9 30              CMP  #"0"
9BCE 90 74              BCC  REN11.     ;KEINE ZIFFER
9BD0 C9 3A              CMP  #:"
9BD2 B0 70              BCS  REN11.     ;KEINE ZIFFER
                ;
                ;ZEILENNUMMER AENDERN:
                ;
9BD4 20 6B A9           JSR  $A96B     ;ZEILENNR -> ADRESSFORMAT (XK)
9BD7 A9 00              LDA  #<TABTAB
9BD9 A2 D6              LDX  #>TABTAB ;ADRESSE REFERENZTABELLE
9BDB 85 AC              STA  ADL
9BDD 86 AD              STX  ADH       ;NACH ADL/ADH
                ;
9BDF 78                 SEI
9BE0 A5 01              LDA  1
9BE2 48                 PHA
9BE3 A9 34              LDA  #$34      ;RAM AUSWAEHLEN
9BE5 85 01              STA  1
                ;
                ;ZEILENNUMMER HINTER BEFEHL
                ;IN REFERENZTABELLE SUCHEN:
                ;
9BE7 38       REN9      SEC
9BE8 A5 AC              LDA  ADL
9BEA E5 FD              SBC  USE      ;USE=AKTUELLER POINTER AUF TABELLE
9BEC A5 AD              LDA  ADH      ;ADRESSE-USE >= 0?
9BEE E5 FE              SBC  USE+1    ;=> ADRESSE >= USE?
9BF0 B0 1E              BCS  REN14    ;JA->FERTIG => NICHT GEFUNDEN
                ;
9BF2 A0 02    REN9.1    LDY  #2
9BF4 B1 AC              LDA  (ADL),Y ;ZEILENNUMMER IN TABELLE
9BF6 C5 14              CMP  XK       ;MIT ZEILENNUMMER VERGLEICHEN
9BF8 D0 07              BNE  REN9.2   ;NICHT GLEICH->WEITER
9BFA C8                 INY
9BFB B1 AC              LDA  (ADL),Y ;HIGH
9BFD C5 15              CMP  XK+1     ;MIT ALTER ZEILENNR. VERGL.
9BFF F0 15              BEQ  REN10    ;GEFUNDEN->ERSETZEN
                ;
```

```
9C01 A5 AC    REN9.2   LDA  ADL
9C03 18                CLC
9C04 69 04             ADC  #4
9C06 85 AC             STA  ADL       ;INC TABELLENZAEHLER
9C08 A5 AD             LDA  ADH
9C0A 69 00             ADC  #0
9C0C 85 AD             STA  ADH
9C0E 90 D7             BCC  REN9      ;UNBEDINGT ->WEITERSUCHEN
              ;
              ;ZEILENNUMMER NICHT GEFUNDEN:
              ;
9C10 A2 F9    REN14    LDX  #$F9      ;DEFAULT-
9C12 A0 FF             LDY  #$FF      ;ZEILENNUMMER=63999
9C14 D0 09             BNE  REN15     ;UNBEDINGT EINSETZEN
              ;
              ;ZEILENNUMMER ERSETZEN:
              ;**********************
              ;
9C16 A0 01    REN10    LDY  #1
9C18 B1 AC             LDA  (ADL),Y ;NEUE ZEILENNUMMER HIGH
9C1A AA                TAX            ;AUS TABELLE
9C1B 88                DEY
9C1C B1 AC             LDA  (ADL),Y ;LOW
9C1E A8                TAY            ;NACH X/Y
              ;
              ;INTEGER->ASCII UMRECHNEN:
              ;
9C1F 68       REN15    PLA
9C20 85 01             STA  1         ;ROM AN
9C22 58                CLI
9C23 86 62             STX  $62       ;VORZEICHEN UEBERBRUECKEN:
9C25 A2 00             LDX  #0
9C27 86 0D             STX  $0D       ;FLAG: NUMERISCH
9C29 84 63             STY  $63       ;ZEILENNUMMER
9C2B A2 90             LDX  #$90      ;VORZEICHEN: PLUS (FLAG)
9C2D 38                SEC
9C2E 20 49 BC          JSR  $BC49     ;ZEILENNR. ($62/$63) NACH FAC
9C31 20 DD BD          JSR  $BDDD     ;FAC NACH ASCII ($100 FF.)
              ;
              ;UND NACH PUFFER UEBERTRAGEN:
              ;
9C34 A6 97             LDX  FLG       ;PUFFERPOINTER HOLEN
9C36 B9 00 01 REN11    LDA  $100,Y    ;ZEICHEN AUS ASCII-PUFFER
9C39 F0 07             BEQ  REN12     ;FERTIG
9C3B E8                INX
9C3C 9D FC 01          STA  $1FC,X    ;NACH ZEILENPUFFER UEBERTRAGEN
9C3F C8                INY
9C40 D0 F4             BNE  REN11     ;UNBEDINGT->NAECHSTES ZEICHEN
              ;
```

```
9C42 86 97    REN12    STX  FLG      ;PUFFERPOINTER WIEDER RETTEN
              ;
              ;SPEZIELLE ZEICHEN AKZEPTIEREN:
              ;
9C44 A2 00    REN11.   LDX  #0
9C46 A1 7A             LDA  (PRGZG,X)  ;LETZTES ZEICHEN HOLEN
9C48 A6 97             LDX  FLG        ;PUFFERZEIGER HOLEN
9C4A C9 2C             CMP  #","     ;KOMMA?
9C4C F0 0B             BEQ  RN8.4.   ;F. ON... NAECHSTE ZEILENNR.
9C4E C9 AB             CMP  #MINUS   ;MINUS?
9C50 F0 07             BEQ  RN8.4.   ;F. LIST...- (NICHT LIST-XXXX!)
9C52 C9 20             CMP  #" "     ;LEERZEICHEN?
9C54 F0 03             BEQ  RN8.4.   ;LEERZEICHEN NICHT IGNORIEREN
9C56 4C 83 9B          JMP  REN6     ;WEITER IM PROGRAMM
              ;
              ;
              ;ZEICHEN NACH PUFFER UEBERTRAGEN:
              ;
9C59 E8       RN8.4.   INX
9C5A 86 97             STX  FLG
9C5C 9D FC 01          STA  $1FC,X   ;NACH PUFFER
9C5F 4C C2 9B          JMP  REN8.4   ;NOCH EINMAL ZEILENNUMMER?
              ;
              ;
              ;
              ;ZEILENENDE BEARBEITEN:
              ;***********************
              ;
9C62 20 FB A8 ENDZEI   JSR  $A8FB    ;Y-REG. ZU PRGZG ADDIEREN
9C65 E8                INX
9C66 8A                TXA
9C67 A8                TAY           ;PUFFERPOINTER IN Y
              ;
9C68 68                PLA           ;AKT. ZEILENNR.-HIGH
9C69 85 15             STA  XK+1
9C6B 68                PLA           ;LOW
9C6C 85 14             STA  XK       ;NACH XK
              ;
9C6E AD 02 03          LDA  EINGAB
9C71 8D 86 C6          STA  ZAHL
9C74 AD 03 03          LDA  EINGAB+1   ;BASIC-VEKTOR FUER
9C77 8D 87 C6          STA  ZAHL+1     ;EINGABEWARTESCHLEIFE RETTEN
9C7A A9 87             LDA  #<REN13
9C7C 8D 02 03          STA  EINGAB
9C7F A9 9C             LDA  #>REN13    ;VEKTOR AUF WEITER
9C81 8D 03 03          STA  EINGAB+1
              ;
              ;ALTE BASIC-ZEILE DURCH ZEILE IM
              ;PUFFER ERSETZEN:
              ;
9C84 4C A2 A4          JMP  $A4A2    ;ERSETZEN DER ALTEN BASIC-ZEILE
```

```
              ;
              ;WEITER:
              ;
9C87 AD 87 C6 REN13    LDA  ZAHL+1
9C8A 8D 03 03          STA  EINGAB+1
9C8D AD 86 C6          LDA  ZAHL
9C90 8D 02 03          STA  EINGAB  ;ALTEN VEKTOR WIEDERHOLEN
              ;
9C93 20 BC A8          JSR  $A8BC   ;SETZT PRGZG AUF START D. BEARB. ZEILE
9C96 A0 05             LDY  #5      ;5
9C98 20 FB A8          JSR  $A8FB   ;ZU PRGZG ADDIEREN
9C9B 20 3B A9          JSR  $A93B   ;ZEILENENDE SUCHEN
              ;
9C9E A0 02    ENDZ1    LDY  #2
9CA0 B1 7A             LDA  (PRGZG),Y  ;PROGRAMMENDE?
9CA2 F0 03             BEQ  ENDZ3      ;JA->FERTIG
9CA4 4C 70 9B          JMP  REN5       ;NAECHSTE ZEILE
              ;
              ;PROGRAMMENDE:
              ;
9CA7 A9 AE    ENDZ3    LDA  #<NEWBRK
9CA9 A0 7B             LDY  #>NEWBRK
9CAB 8D 18 03          STA  IRQ+4
9CAE 8C 19 03          STY  IRQ+5   ;ALTER NMI-VEKTOR
9CB1 20 59 A6          JSR  $A659   ;CLR-BEF.
9CB4 20 33 A5          JSR  $A533   ;ZEILEN BINDEN
9CB7 4C 85 E3          JMP  $E385   ;WARMSTART
              ;
              ;
              ;TABELLE DER TOKENS FUER:
              ;
              ;GOSUB/GOTO/THEN/RUN/LIST
              ;(BEFEHLE MIT ZEILENNUMMERN)
              ;
9CBA 8D 89 A7 RENDAT   .BYT $8D,$89,$A7,$8A,$9B
              ;
              ;
              ;UNTERROUTINE ZUR DURCHFUEHRUNG
              ;DER ERSTEN ZWEI PASSES:
              ;*******************************
              ;
9CBF A9 00    PASSES   LDA  #<TABTAB
9CC1 A2 D6             LDX  #>TABTAB
9CC3 85 FD             STA  USE
9CC5 86 FE             STX  USE+1   ;TABELLE DER ZEILENNUMMERN (RAM)
              ;
9CC7 78                SEI
9CC8 A5 01             LDA  1
9CCA 48                PHA
9CCB A9 34             LDA  #$34    ;RAM AUSWAEHLEN
9CCD 85 01             STA  1
```

```
                ;
9CCF A5 2B              LDA  BASSTA
9CD1 A6 2C              LDX  BASSTA+1  ;ZEIGER AUF BASICSTART
9CD3 85 5F      PASS1   STA  $5F
9CD5 86 60              STX  $60       ;NACH $5F/$60
9CD7 A0 01              LDY  #$01
9CD9 B1 5F              LDA  ($5F),Y   ;LINK-ADR.-HIGH
9CDB F0 44              BEQ  PASS2     ;FERTIG (PROGRAMMENDE)
                ;
9CDD C8                 INY            ;Y=2
9CDE B1 5F              LDA  ($5F),Y   ;ZEILENNUMMER-LOW
9CE0 91 FD              STA  (USE),Y   ;ALS ALTE NR. SPEICHERN
9CE2 C8                 INY            ;Y=3
9CE3 B1 5F              LDA  ($5F),Y   ;HIGH
9CE5 91 FD              STA  (USE),Y
                ;
9CE7 A5 15              LDA  XK+1    ;NEUE HIGH
9CE9 C9 FA              CMP  #$FA    ;MAX. 63999
9CEB B0 3E              BCS  RENU3   ;FEHLER
9CED 48                 PHA
9CEE 24 96              BIT  FLG2    ;PASS-FLAG
9CF0 30 07              BMI  RENU2   ;ZEILENNUMMER NUR TESTEN
                ;
9CF2 91 5F              STA  ($5F),Y ;ZEILENNUMMER AENDERN
9CF4 88                 DEY          ;Y=2
9CF5 A5 14              LDA  XK      ;NEUE LOW
9CF7 91 5F              STA  ($5F),Y ;AENDERN (IM PROGRAMM)
                ;
9CF9 A0 00      RENU2   LDY  #0      ;Y=0
9CFB A5 14              LDA  XK
9CFD 91 FD              STA  (USE),Y ;NEUE IN TABELLE (LOW)
9CFF 18                 CLC
9D00 65 97              ADC  FLG
9D02 85 14              STA  XK      ;ZEILENNUMMER ERHOEHEN (LOW)
9D04 C8                 INY          ;Y=1
9D05 68                 PLA
9D06 91 FD              STA  (USE),Y ;HIGH
9D08 69 00              ADC  #0
9D0A 85 15              STA  XK+1    ;HIGH-BYTE ERHOEHEN
9D0C B0 1D              BCS  RENU3   ;ZEILENNUMMER ZU HOCH (C=1)
                ;
9D0E A5 FD              LDA  USE
9D10 69 04              ADC  #4
9D12 85 FD              STA  USE     ;TABELLENADR. INC
9D14 90 02              BCC  REN4
9D16 E6 FE              INC  USE+1
9D18 B1 5F      REN4    LDA  ($5F),Y ;ADRESSE DER NAECHSTEN ZEILE
9D1A AA                 TAX          ;LOW
9D1B 88                 DEY          ;Y=0
9D1C B1 5F              LDA  ($5F),Y ;HIGH
9D1E 4C D3 9C           JMP  PASS1
```

```
               ;
               ;PROGRAMMENDE:
               ;
9D21 A0 01     PASS2     LDY  #1
9D23 A9 FF               LDA  #$FF
9D25 91 FD               STA  (USE),Y ;ENDE-ZEICHEN: 2X$FF
9D27 C8                  INY
9D28 91 FD               STA  (USE),Y ;NACH TABELLE
9D2A 18                  CLC          ;OK-FLAG
               ;
9D2B 68        RENU3     PLA
9D2C 85 01               STA  1       ;ROM AUSWAEHLEN
9D2E 58                  CLI          ;INTERRUPT EIN
9D2F 60                  RTS          ;ZURUECK: ;C=0: OK // C=1: FEHLER
               ;
               ;
               ;
               ;********************
               ;**                **
               ;**  BEFEHL: POS=  **
               ;**                **
               ;********************
               ;
               ;
9D30 20 9E B7  POSITI    JSR  GETBYT  ;SPALTE HOLEN
9D33 E0 28               CPX  #40
9D35 90 02               BCC  POS1
9D37 A2 27               LDX  #39     ;KORREKTUR
9D39 86 D3     POS1      STX  $D3     ;CURSORSPALTE
               ;
9D3B 20 F1 B7            JSR  CHKGET  ;ZEILE HOLEN
9D3E E0 19               CPX  #25
9D40 90 02               BCC  POS2
9D42 A2 18               LDX  #24     ;KORREKTUR
9D44 86 D6     POS2      STX  $D6     ;ZEILE
9D46 4C 6C E5            JMP  $E56C   ;CURSOR SETZEN
               ;
               ;
               ;
```

```
              ;*******************
              ;**               **
              ;**  BEFEHL: KEY  **
              ;**               **
              ;*******************
              ;
              ;
9D49 20 9E B7 KEY       JSR  GETBYT  ;FUNKTIONSTASTE (1-8)->X
9D4C CA                 DEX          ;AUS 1-8 MACH 0-7
9D4D 8A                 TXA
9D4E 29 07              AND  #$07    ;KORREKTUR
9D50 0A                 ASL  A
9D51 0A                 ASL  A
9D52 0A                 ASL  A
9D53 0A                 ASL  A       ;* 16
9D54 48                 PHA          ;POINTER
              ;
9D55 20 FD AE           JSR  CHKCOM  ;AUF KOMMA TESTEN
9D58 20 9E AD           JSR  FRMEVL  ;STRINGPARAMETER (LAENGE->AKU)
9D5B 20 A3 B6           JSR  FRESTR  ;STRINGVERWALTUNG
9D5E 85 97              STA  FLG     ;LAENGE
9D60 A2 10              LDX  #16     ;MAX. 16 BUCHSTABEN
9D62 86 AE              STX  SHP
9D64 68                 PLA
9D65 AA                 TAX          ;POINTER
              ;
              ;STRING UEBERTRAGEN:
              ;
9D66 A0 00              LDY  #0
9D68 B1 22    KEY1      LDA  ($22),Y   ;HOLE STRINGBYTE
9D6A 9D 8F C7           STA  FUNKTI,X  ;FUNKTION UEBERTRAGEN
9D6D C8                 INY
9D6E E8                 INX
9D6F C6 97              DEC  FLG
9D71 D0 0B              BNE  KEY3
9D73 A9 A0              LDA  #$A0    ;REST MIT $A0 AUFFUELLEN
9D75 9D 8F C7 KEY2      STA  FUNKTI,X
9D78 E8                 INX
9D79 C6 AE              DEC  SHP
9D7B D0 F8              BNE  KEY2
9D7D 60                 RTS
              ;
              ;
9D7E C6 AE    KEY3      DEC  SHP
9D80 D0 E6              BNE  KEY1
9D82 60       KEY4      RTS
              ;
              ;
              ;
```

```
                  ;*************************
                  ;**                     **
                  ;**  BEFEHL: DIRECTORY  **
                  ;**                     **
                  ;*************************
                  ;
                  ;
9D83 A2 08        DIREKT  LDX  #$08      ;GERAETEADRESSE
9D85 A0 00                LDY  #0        ;SECUNDAERADRESSE
9D87 20 BA FF             JSR  $FFBA     ;FILEPARAMETER SETZEN
9D8A 20 F2 8E             JSR  SRCHFN    ;FILENUMMER SETZEN
9D8D A9 01                LDA  #1          ;LAENGE
9D8F A2 DB                LDX  #<DOLLAR
9D91 A0 9D                LDY  #>DOLLAR  ;ADRESSE FUER FILENAME
9D93 20 BD FF             JSR  $FFBD       ;FILENAMENPARAMETER
9D96 20 C1 E1             JSR  OPEN      ;DATEI OEFFNEN
9D99 A6 B8                LDX  FILENR    ;FILENUMMER
9D9B 20 1E E1             JSR  CHKIN     ;EINGABEGERAET SETZEN
                  ;
9D9E 20 12 E1             JSR  BASIN     ;ZEICHEN VON DISK
9DA1 20 12 E1             JSR  BASIN     ;ERSTE 2 ZEICHEN AUSLASSEN
9DA4 4C C4 9D             JMP  DIR0
                  ;
                  ;HAUPTSCHLEIFE:
                  ;
9DA7 20 12 E1     DIR1    JSR  BASIN
9DAA 85 FD                STA  USE
9DAC 20 12 E1             JSR  BASIN
9DAF A6 FD                LDX  USE       ;HOLE SEKTORENZAHL->A/X
9DB1 20 CD BD             JSR  INTOUT    ;INTEGERZAHL IN A/X AUSGEBEN
9DB4 A9 20                LDA  #$20      ;SPACE
9DB6 20 0C E1             JSR  BASOUT    ;AUSGEBEN
                  ;
9DB9 20 12 E1     DIR2    JSR  BASIN     ;ZEICHEN HOLEN
9DBC F0 06                BEQ  DIR0      ;ZEILENENDE
9DBE 20 0C E1             JSR  BASOUT    ;AUSGEBEN
9DC1 4C B9 9D             JMP  DIR2      ;NAECHSTES ZEICHEN
                  ;
                  ;
9DC4 A9 0D        DIR0    LDA  #$0D      ;CARRIAGE RETURN
9DC6 20 0C E1             JSR  BASOUT    ;ZEICHEN IN A AUSGEBEN
9DC9 20 12 E1             JSR  BASIN
9DCC 20 12 E1             JSR  BASIN     ;ERSTEN ZWEI BYTES UNNOETIG
                  ;
9DCF 20 E1 FF             JSR  CHKSTP    ;PRUEFE AUF STOP-TASTE
9DD2 F0 04                BEQ  DIR3      ;BEI STOP -> ENDE
9DD4 A5 90                LDA  STATUS    ;FEHLER ODER DATEI ENDE?
9DD6 F0 CF                BEQ  DIR1      ;WEITER
                  ;
```

```
               ;ENDE:
               ;
9DD8 4C 69 8E DIR3     JMP  GL2     ;CLRCHANNEL+CLOSE
               ;
               ;
               ;OPEN-FILENAME:
               ;
9DDB 24        DOLLAR   .ASC "$"
               ;
               ;
               ;
               ;
               ;
               ;*********************
               ;**                 **
               ;**  BEFEHL: PAINT  **
               ;**                 **
               ;*********************
               ;
               ;
9DDC 20 67 7C PAINT    JSR  STFLG   ;ZEICHENFLAG HOLEN
9DDF 20 FF 7F          JSR  TESCOR  ;KOORDINATEN HOLEN
9DE2 48                PHA
9DE3 A5 02             LDA  GFLAG
9DE5 4A                LSR  A
9DE6 B0 0D             BCS  PAI1    ;LGR
9DE8 68                PLA
9DE9 20 AA 81          JSR  HPOSN   ;PUNKTADRESSE BERECHNEN
9DEC 20 63 88          JSR  GETE5   ;E5 BERECHNEN
9DEF 8D 15 C6          STA  E5.ZW
9DF2 4C F7 9D          JMP  PAI2    ;WEITER
               ;
9DF5 68        PAI1    PLA
9DF6 60                RTS          ;PAINT NICHT FUER LGR!
               ;
9DF7 A9 00     PAI2    LDA  #0
9DF9 85 58             STA  STKPOI  ;STACKPOINTER AUF NULL
9DFB 8D 91 C6          STA  RUECK   ;RUECKMELDEREGISTER AUF NULL
               ;
9DFE 20 AA 9E          JSR  PUTSTK  ;STARTPUNKT AUF STACK
               ;
9E01 20 4C 9E          JSR  TESTE   ;BEI X,Y PUNKT?
9E04 84 57             STY  DO      ;ZUSTAND MERKEN
               ;
               ;
```

```
              ;GROSSE SCHLEIFE (FELD AUSFUELLEN):
              ;*********************************
              ;
              ;RAND SUCHEN
9E06
              ;
9E06 20 78 9E PAI3     JSR  SAVREG  ;PUNKTADRESSE MERKEN
9E09 20 9A 9E          JSR  LINKS.  ;X-1
9E0C 20 4C 9E          JSR  TESTE   ;PUNKT BEI X-1,Y TESTEN
9E0F C4 57             CPY  D0      ;RAND?
9E11 F0 F3             BEQ  PAI3    ;JA!->ADRESSE ALS NEUE ADR. MERKEN
              ;
              ;LINIE NACH RECHTS ZEICHNEN
9E13
              ;
9E13 20 89 9E PAI5     JSR  GETREG  ;AKTUELLE ADRESSE HOLEN
9E16 20 FA 86          JSR  UNTEN   ;Y+1
9E19 20 FD 9E          JSR  TEST1   ;TESTE PUNKT X,Y+1
              ;
9E1C 20 35 87          JSR  OBEN    ;Y-1
9E1F 20 FD 9E          JSR  TEST1   ;TESTE PUNKT X,Y-1
              ;
9E22 20 18 82          JSR  PLT     ;PUNKT SETZEN
9E25 4C 37 9E          JMP  PAI7    ;NAECHSTEN PUNKT
              ;
              ;
9E28 20 18 82 PAI6     JSR  PLT     ;PUNKT SETZEN
9E2B 20 35 87          JSR  OBEN    ;Y-1
9E2E 20 0A 9F          JSR  TEST2
              ;
9E31 20 FA 86          JSR  UNTEN   ;Y+1
9E34 20 0A 9F          JSR  TEST2
              ;
9E37 20 A2 9E PAI7     JSR  RECHT.  ;X+1
9E3A 20 78 9E          JSR  SAVREG  ;NAECHSTEN PUNKT RECHTS MERKEN
9E3D 20 4C 9E          JSR  TESTE   ;RAND?
9E40 C4 57             CPY  D0
9E42 F0 E4             BEQ  PAI6    ;NEIN->WEITER ZEICHNEN
              ;
              ;JA:
              ;
9E44 20 D4 9E          JSR  GETSTK  ;ADRESSE VON STACK HOLEN
9E47 90 BD             BCC  PAI3    ;AB DIESER ADRESSE WEITERZEICHNEN
9E49 4C C4 7F          JMP  PLTVAR  ;FERTIG
              ;
              ;
              ;
```

```
              ;AUF PUNKT TESTEN:
              ;*****************
              ;
9E4C 78       TESTE    SEI
9E4D A5 01             LDA  1
9E4F 48                PHA
9E50 29 FC             AND  #$FC
9E52 85 01             STA  1          ;ROM AUS
              ;
9E54 A0 00             LDY  #0
9E56 A5 AB             LDA  MSK        ;MASKE HOLEN
9E58 31 AC             AND  (ADL),Y    ;PUNKT HOLEN
              ;
9E5A A0 08             LDY  #8
9E5C 4A       TESTE2   LSR  A
9E5D B0 03             BCS  TESTE3
9E5F 88                DEY
9E60 D0 FA             BNE  TESTE2
9E62 2A       TESTE3   ROL  A
              ;
9E63 A8                TAY
              ;
9E64 AD 91 C6          LDA  RUECK     ;AUSSERHALB BILDSCHIRM?
9E67 F0 0A             BEQ  TESTE1    ;NEIN!
              ;
9E69 A5 57             LDA  D0
9E6B 49 FF             EOR  #$FF
9E6D A8                TAY            ;BILDSCHIRMRAND ALS BEGRENZUNG!
9E6E A9 00             LDA  #0
9E70 8D 91 C6          STA  RUECK     ;RUECKMELDUNG LOESCHEN
              ;
9E73 68       TESTE1   PLA
9E74 85 01             STA  1
9E76 58                CLI            ;ROM EIN
              ;
9E77 60                RTS
              ;
              ;
              ;AKTUELLE PUNKTADRESSE SPEICHERN:
              ;*********************************
              ;
9E78 A5 AC    SAVREG   LDA  ADL
9E7A 85 59             STA  D2
9E7C A5 AD             LDA  ADH
9E7E 85 5A             STA  D3
9E80 A5 AB             LDA  MSK
9E82 85 5B             STA  D4
9E84 A5 93             LDA  E5
9E86 85 5C             STA  D5
9E88 60                RTS
              ;
```

```
               ;
               ;AKTUELLE PUNKTADRESSE HOLEN:
               ;*****************************
               ;
9E89 A5 59     GETREG   LDA  D2
9E8B 85 AC              STA  ADL
9E8D A5 5A              LDA  D3
9E8F 85 AD              STA  ADH
9E91 A5 5B              LDA  D4
9E93 85 AB              STA  MSK
9E95 A5 5C              LDA  D5
9E97 85 93              STA  E5
9E99 60                 RTS
               ;
               ;
               ;GRAPHIKCURSOR LINKS+RECHTS:
               ;****************************
               ;
9E9A A4 93     LINKS.   LDY  E5
9E9C 20 B4 87           JSR  LINKS
9E9F 84 93              STY  E5
9EA1 60                 RTS
               ;
               ;
9EA2 A4 93     RECHT.   LDY  E5
9EA4 20 80 87           JSR  RECHTS
9EA7 84 93              STY  E5
9EA9 60                 RTS
               ;
               ;
               ;AKTUELLE ADRESSE AUF STACK:
               ;****************************
               ;
9EAA A4 58     PUTSTK   LDY  STKPOI   ;STACKPOINTER LADEN
               ;
9EAC 78                 SEI
9EAD A5 01              LDA  1
9EAF 48                 PHA
9EB0 29 FC              AND  #$FC
9EB2 85 01              STA  1        ;ROM AUS
               ;
9EB4 A5 AC              LDA  ADL
9EB6 99 00 D6           STA  STADL,Y
9EB9 A5 AD              LDA  ADH
9EBB 99 00 D7           STA  STADH,Y
9EBE A5 AB              LDA  MSK
9EC0 99 00 D8           STA  STMSK,Y
9EC3 A5 93              LDA  E5
9EC5 99 00 D9           STA  STE5,Y
               ;
```

```
9EC8 68                PLA
9EC9 85 01             STA  1
9ECB 58                CLI            ;ROM AN
                ;
9ECC E6 58             INC  STKPOI    ;STACKPOINTER + 1
9ECE F0 01             BEQ  ERRO      ;FEHLER
9ED0 60                RTS
                ;
9ED1 4C 35 A4   ERRO   JMP  $A435     ;OUT OF MEMORY
                ;
                ;
                ;AKTUELLE ADRESSE VON STACK:
                ;****************************
                ;
9ED4 38         GETSTK SEC            ;FLAG FUER STACKENDE
9ED5 C6 58             DEC  STKPOI    ;STACKPOINTER - 1
9ED7 F0 23             BEQ  RTS..     ;FERTIG
                ;
9ED9 A4 58             LDY  STKPOI    ;STACKPOINTER LADEN
                ;
9EDB 78                SEI
9EDC A5 01             LDA  1
9EDE 48                PHA
9EDF 29 FC             AND  #$FC
9EE1 85 01             STA  1         ;ROM AUS
                ;
9EE3 B9 00 D6          LDA  STADL,Y
9EE6 85 AC             STA  ADL
9EE8 B9 00 D7          LDA  STADH,Y
9EEB 85 AD             STA  ADH
9EED B9 00 D8          LDA  STMSK,Y
9EF0 85 AB             STA  MSK
9EF2 B9 00 D9          LDA  STE5,Y
9EF5 85 93             STA  E5
                ;
9EF7 68                PLA
9EF8 85 01             STA  1
9EFA 58                CLI            ;ROM AN
                ;
9EFB 18                CLC            ;FLAG FUER OK.
                ;
9EFC 60         RTS..  RTS
                ;
                ;
```

```
              ;ARBEITSUPRO 1 - TESTE MIT MERKEN:
              ;**********************************
              ;
9EFD 20 4C 9E TEST1    JSR  TESTE   ;TESTE PUNKT
9F00 C4 57             CPY  DO      ;RAND?
9F02 D0 03             BNE  TES1    ;JA!
9F04 20 AA 9E          JSR  PUTSTK  ;PUNKT AUF STACK
9F07 4C 89 9E TES1     JMP  GETREG  ;AKTUELLE ADRESSE HOLEN
              ;
              ;
              ;ARBEITSUPRO 2 - TESTE 2 MIT MERKEN:
              ;************************************
              ;
9F0A 20 4C 9E TEST2    JSR  TESTE   ;PUNKT TESTEN
9F0D C4 57             CPY  DO      ;RAND?
9F0F D0 2C             BNE  TES3    ;JA!
              ;
9F11 A5 AC             LDA  ADL
9F13 48                PHA
9F14 A5 AD             LDA  ADH
9F16 48                PHA
9F17 A5 AB             LDA  MSK
9F19 48                PHA
9F1A A5 93             LDA  E5
9F1C 48                PHA          ;ADRESSE MERKEN
              ;
9F1D 20 9A 9E          JSR  LINKS.  ;X-1
9F20 20 4C 9E          JSR  TESTE   ;PUNKT TESTEN
9F23 C4 57             CPY  DO      ;RAND?
9F25 F0 12             BEQ  TES2    ;NEIN!
              ;
9F27 68                PLA
9F28 85 93             STA  E5
9F2A 68                PLA
9F2B 85 AB             STA  MSK
9F2D 68                PLA
9F2E 85 AD             STA  ADH
9F30 68                PLA
9F31 85 AC             STA  ADL     ;ADRESSE HOLEN
              ;
9F33 20 AA 9E          JSR  PUTSTK  ;UND AUF STACK
9F36 4C 89 9E          JMP  GETREG  ;AKTUELLE KOORDINATEN HOLEN
              ;
9F39 68       TES2     PLA
9F3A 68                PLA
9F3B 68                PLA
9F3C 68                PLA          ;STACK AUFRAEUMEN
9F3D 4C 89 9E TES3     JMP  GETREG  ;AKTUELLE KOORDINATEN HOLEN
              ;
```

Und hier die Symboltabelle:

ABS	C610	ADH	00AD	ADL	00AC
ADRTAB	7B5F	ADRZW	C64B	AT	0091
B0	7A29	B1	7A7B	B3	7A6A
BASIN	E112	BASOUT	E10C	BASSTA	002B
BC1	8BFB	BCOL	C619	BCOLO.	8BE8
BCOLOR	8BDF	BEFEHL	7A20	BEFO	7A88
BEFO1	7A8E	BEFTAB	7A93	BGRDZW	C616
BIPO	86F9	BIPTB1	86C8	BIPTB2	86D0
BRK0	7BC6	BRK1	7BD4	BRKGET	7BD2
BRKSAV	C62C	C1	C656	C2	C657
C3	C658	C4	C659	C5	C65A
C6	C65B	C7	C65C	CCONT	9A33
CEXIT	9A89	CFLAG	C68B	CHKCOM	AEFD
CHKGET	B7F1	CHKIN	E11E	CHKOUT	E118
CHKSTP	FFE1	CHKSTR	AD8F	CHRGET	0073
CHRGOT	0079	CI4	99A9	CI5	99EE
CI6	99F6	CIRC	9837	CIRC0	983F
CIRC1	984D	CL101	9978	CL102	99AD
CL103	99B9	CL2	98A6	CL3	98B5
CL4	98B2	CLOSE	E1CC	CLQ1	9A66
CLQW	98C8	CLRCH	FFCC	CLRCOL	92E1
CMODE	C65E	CMOVE	9A2B	CMUL	991B
CMUL1	997B	CMUL16	9952	CMUL17	9929
CNEGX	99FF	CNEGY	99AF	CNEXT	9A28
CNEXT5	98E0	CNEXTQ	9A8A	COL1	C61A
COL2	C61B	COL3	C61C	COL3ZW	C675
COLMEM	C650	COLMM	C653	COLOR	C655
COLTB1	82CC	COMBT2	8509	COMBTB	8505
CPSEX	9A72	CRDYX	9A14	CRDYY	99BB
CSTAB	9AAB	CSUB	98FE	CSUB1	98F0
CW1	C661	CW2	C663	CXC	C668
CXG	C665	CYC	C66A	CYG	C660
D.0	8D0D	D.1	8D40	D.2	8D17
D.3	8D24	D.4	8D31	D.5	8D3C
D.6	8D46	D0	0057	D1	0058
D2	0059	D3	005A	D4	005B
D5	005C	D6	005D	D7	005E
D8	8CBE	D9	8CC1	DA	8D57
DATAST	9B1C	DCOLTB	C699	DE2	8F40
DE2.	8F6C	DE3	8F7B	DE4	8F63
DEFINE	8F10	DFABTB	7847	DI0	8DFE
DI1	8DCE	DI2	8DBC	DI2.1	8E26
DI2.2	8E19	DI3	8DB1	DI4	8DAB
DI5	8DB9	DI7	8DE4	DI8	8DDD
DI9	8DEF	DIR0	9DC4	DIR1	9DA7
DIR2	9DB9	DIR3	9DD8	DIREKT	9D83
DISK	8DA3	DISK2	8DFF	DOLLAR	9DDB
DR1	8C9E	DR2	8CC7	DR3	8CE5

DRAW	8C7E	DRAW.1	8CEE	DRAW2	8D4B
DRAW3	8D4C	DRAW4	8D65	E0	C610
E1	C611	E2	C612	E5	0093
E5.ZW	C615	E7	C613	EINGAB	0302
ENDZ1	9C9E	ENDZ2	9B9B	ENDZ3	9CA7
ENDZEI	9C62	ERRO	9ED1	FABMEM	C651
FABMM	C654	FABRSC	8A53	FACINT	B7F7
FENSTR	C6AA	FI0	809E	FI1	8198
FI2	81A6	FI3	80C3	FIL0	96D0
FIL1	96B1	FILENR	00B8	FILL	809B
FILON	C670	FILTER	9690	FINIT	8155
FINITO	8158	FLAGP1	C692	FLAGP2	C693
FLG	0097	FLG2	0096	FR0	80DA
FR3	8152	FR7	8145	FR8	8138
FR9	8124	FRAD1H	0065	FRAD1L	0064
FRAD2H	006E	FRAD2L	006D	FRAME	80C6
FRESTR	B6A3	FRFILL	8B36	FRMEVL	AD9E
FRMNUM	AD8A	FSTBEF	0057	FUNKTI	C78F
FUNVOR	780A	GC1	8375	GC2	8378
GC3	8384	GC4	83A5	GC5	83BA
GC6	83D7	GCFLAG	00AE	GCLEAR	8362
GCLV	83B3	GCO0	8454	GCO2	845E
GCO3	8465	GCO4	844D	GCO5	84F3
GCO6	84F5	GCOAND	84A8	GCOCOP	84BF
GCOEND	84A5	GCOEOR	84B8	GCOJ	8493
GCOMB	841E	GCOOR	84B1	GCOV	84DF
GEND	C617	GET	95F0	GET1	9257
GET15	95ED	GETBYT	B79E	GETCOR	B7EB
GETE5	8863	GETE5.	8871	GETGR1	9AAA
GETGR2	9AA8	GETGRA	9A91	GETNR	9247
GETPAR	E1D4	GETRA1	8C76	GETRAS	8C64
GETREG	9E89	GETSTK	9ED4	GETSTR	8EFE
GETTAB	8D9F	GETZ1	7C41	GETZ2	7C15
GETZ3	7C1E	GETZ6	7C14	GETZ7	7C35
GETZ8	7C3E	GETZ9	7C3B	GETZA	7BFC
GETZEI	7BE4	GFLAG	0002	GFLAG2	C61E
GL0	8ED9	GL0.	8EEC	GL00	8ED7
GL1	8E61	GL2	8E69	GL2.	8E68
GL3	8E71	GLOAD	8E4B	GLOAD.	8EB9
GM00	8A86	GM000	8A75	GM01	8AA2
GM1	8AAB	GM12	8C38	GM13	8C42
GM2	8AB0	GM2A	8AC1	GM2B	8AC9
GM2C	8ACD	GM2C.	8AE0	GM2D	8AF4
GM2D.	8AEE	GM3	8B22	GM5	8B44
GM9	8C1D	GM9.	8C32	GMODE	8A8B
GMODEI	8A6C	GMOVE	8874	GMTAB	8A5F
GMTAB2	8A68	GMV0	88B6	GMV1	88AD
GMV2	8906	GMV3	8922	GMV4	8925
GMV5	8911	GMV7	8895	GMV8	8889
GMV9	88D7	GOTO	A8A3	GRAD1H	0061
GRAD1L	0060	GRAD2H	006A	GRAD2L	0069

GRAPHM	8AFA	GRAPM1	8AFD	GRI0	86A5
GRI1	867F	GRI2	868D	GRI3	8698
GRI4	86A2	GRI5	86A4	GRIGHT	8657
GRMEM	C64F	GRMM2	C652	GROFF	7C59
GS0	8E92	GS0.	8EB3	GS1	8E3E
GSAVE	8E28	GSAVE.	8E77	GSTART	C618
H6	8205	H7	8208	HARD	C400
HKFLG1	000F	HKFLG2	0008	HL4	881C
HLINE	87E4	HPOSN	81AA	HPS1	81BB
I0	7757	I0.	7795	I0..	7712
I0...	772C	I00	7728	I00.	774B
I00..	773C	I00...	773F	I000	7747
I1	77EB	I1.	7702	I10	79A1
I10.	799F	I11	79BA	I12	79B1
I2	77F6	I3	78F7	I3.A	7912
I3.B	791C	I4	7935	I4.2	7933
I4.A	7948	I4.B	794A	I4.C	7951
I5	795A	I6	7961	I6.A	7965
I7	7975	I8	791E	I9	7985
IFZ	91C8	IFZ1	91CA	IFZ2	91D5
IFZ3	91F9	IFZ4	91FC	IFZ5	9208
IFZ6	9213	IFZ7	9217	IFZ8	9228
IFZDAT	91C2	III	77E0	ILLFE	8F0D
INIT	7700	INTOUT	BDCD	INTPO	79C1
INTPRT	78F1	INV1	83ED	INV2	83F0
INV3	83FC	INV4	840E	INVERS	83DA
INVV	8409	IPL	82C5	IRQ	0314
IRQ0	93F3	IRQ1	9413	IRQ2	945A
IRQ3	943C	IRQNEU	93D7	IRQSAV	C62A
IRQWAR	93F5	IRRSAV	C68A	JLINKS	877D
JMPSCL	8B96	JOBEN	8777	JRECHT	877A
JUMPTI	9466	JUNTEN	8774	KEY	9D49
KEY1	9D68	KEY2	9D75	KEY3	9D7E
KEY4	9D82	KOPF	7857	L.VFLG	0093
L1	8831	L1.	8847	L10	7E56
L11	7E4D	L12	7E6E	L12D	7E96
L12E	7EBC	L12F	7EA0	L13	7E5D
L16	7EF0	L2	8840	L3	8075
L4	8801	L5	8836	L6	884F
L7	7E0A	L8	7E17	L9	7E6C
LCOLO	C61F	LGMV0	8934	LGMV1	896A
LGMV2	8971	LGMV3	895B	LGMV4	8953
LGRMOV	892F	LI1	87BD	LI2	87DA
LI3	87DB	LINE.	805E	LINKS	87B4
LINKS.	9E9A	LIPLOT	7EEA	LIS0	79D5
LIS1	79DD	LIS2	79E0	LIS3	79E8
LIS4	79F3	LIS5	79C9	LIST	79C4
LIVER	8974	LLINE	7F00	LLINE.	807E
LLINKS	7F56	LLK1	7F5E	LO1	7F44
LOBEN	7F3C	LODOUT	F5D2	LPL2	7E8D
LPL4	7E78	LPL5	7E7C	LPLOT.	8054

LPLT	7DF2	LPLTHP	7DF2	LPLTT	7EE2
LR1	7F52	LRECHT	7F48	LRV1	8908
LSTO	79C6	LU1	7F38	LUNTEN	7F2E
LV1	8976	LV2	8980	LV3	8996
LV4	899E	LV5	89B0	LV6	89C9
LV7	89DF	LV8	89C7	LVC	89BC
LZW	C649	MCSC	8A43	MCSC1	8A4D
MERGE	9B05	MINUS	00AB	MSK	00AB
MSKHG	8C5C	MSKMC	8C60	MSKTB1	7D27
MSKTB2	7D2F	MUL.H	7D37	MUL.L	7D51
NEWBRK	7BAE	NLPL	7EBE	NPL	827A
NPL2	827D	NR	C64E	OB1	8744
OB2	8753	OB3	8765	OB4	8769
OBEN	8735	OERR	B97E	OFFS	8F7C
OFFS1	8F88	OFFX	006D	OPEN	E1C1
P1	7FE6	P2	7FF0	PA1C	00C0
PA1F	00D8	PA1G	00E0	PA2C	00C8
PA2F	00CC	PA2G	00A0	PAGE1C	78EC
PAGE1F	78ED	PAGE1G	78EB	PAGE2C	78EF
PAGE2F	78F0	PAGE2G	78EE	PAI1	9DF5
PAI2	9DF7	PAI3	9E06	PAI5	9E13
PAI6	9E28	PAI7	9E37	PAINT	9DDC
PASS1	9CD3	PASS2	9D21	PASSES	9CBF
PCL1	8BA2	PCOLHG	8BBC	PCOLMC	8BCA
PCOLOR	8B99	PDL	911A	PDL1	912D
PDL2	913B	PDL3	9149	PFLAGS	C6F9
PINSEL	C690	PL0	822E	PL1	824A
PL2	8281	PL3	8290	PL4	823C
PL5	8240	PLOT	7FD9	PLOT.	8047
PLOTMD	7D55	PLT	8218	PLTT	82BC
PLTVA.	7FCA	PLTVAR	7FC4	POITAB	96F0
POS1	9D39	POS2	9D44	POSITI	9D30
PRGZG	007A	PRGZGR	C68E	PUTSTK	9EAA
QERR	B248	R.L	87B2	RASFLG	C694
RASSPR	941F	RAST1	C637	RAST2	C638
RDAT	8D68	RE1	8789	RE2	87A8
RE3	87A9	RECHT.	9EA2	RECHTS	8780
REGTAB	91AC	REN10	9C16	REN11	9C36
REN11.	9C44	REN12	9C42	REN13	9C87
REN14	9C10	REN15	9C1F	REN4	9D18
REN5	9B70	REN6	9B83	REN7	9B85
REN8	9BAE	REN8.0	9B9E	REN8.1	9BBF
REN8.2	9BB2	REN8.4	9BC2	REN8.5	9BC8
REN9	9BE7	REN9.1	9BF2	REN9.2	9C01
RENDAT	9CBA	RENU2	9CF9	RENU3	9D2B
RENUM	9B39	RENUM.	9B42	RET1	7A10
RETIRQ	79F6	REVER	89E0	RN8.4.	9C59
ROTCOL	8C43	RSFLG1	C695	RT	C614
RTS.	8A8A	RTS..	9EFC	RUECK	C691
RV1	89EC	RV2	89F8	RV3	89FA
RV4	8A04	RV5	8A11	RV6	8A35

RV7	8A28	RV8	8A33	RVC	8A1D
S	D400	SAD	C65D	SAVOUT	F68F
SAVREG	9E78	SCALE	8D79	SCLHG1	C6E7
SCLHG2	C6F1	SCLMC1	C6E5	SCLMC2	C6EF
SCOL	8C09	SCOL1	8C0D	SCOL3	8B72
SCOLOR	8B69	SECADR	00B9	SEITTA	84D0
SERR	AF08	SERR1	81A7	SETCUR	7FB0
SHP	00AE	SI0	9500	SI1	9488
SI2	9490	SI3	94BD	SI4	94E6
SI5	94D5	SI6	94DB	SI7	94BA
SIRQ	9469	SIRQGO	93F0	SKORD1	C6C4
SKORD2	C6D5	SL9	7FAC	SLCOLO	8BD8
SLD1	9396	SLD2	938F	SLD3	9392
SLD4	93A1	SLDCOR	937C	SLINKS	7F94
SLOW	C630	SM0	907E	SM1	9092
SM1.	9095	SM2	9084	SM5	909F
SM6	90A0	SM7	90A8	SM8	90F0
SM9	90CC	SMODE	9048	SOBEN	7F6C
SOFF16	93AE	SOFF17	93BB	SOU1	9561
SOU2	9584	SOU3	95B4	SOU5	95BB
SOU6	95BE	SOUND	954E	SPLOT	820B
SPMC1	C72B	SPMC2	C72C	SPON1	C729
SPON2	C72A	SPPRI1	C72D	SPPRI2	C72E
SPRDEF	00D2	SPRET	8646	SPRI16	C6AB
SPRIT1	C688	SPRIT2	C689	SPRPOW	90F1
SPVEK	C719	SPXEX1	C731	SPXEX2	C732
SPYEX1	C72F	SPYEX2	C730	SR0	8FE6
SR1	8FAF	SR2	8FBF	SR3	8FDB
SR4	8FD0	SR5	8FF9	SR6	9013
SR7	902B	SR9	7F90	SRCHFN	8EF2
SRCHOU	F5AF	SRE1	9181	SREAD	8F93
SRECHT	7F76	SREG	D000	SREGIN	915F
SREGSG	9162	SRUNS1	C66E	SRUNS2	C66F
SS0	9292	SS1	926C	SS3	92C0
SS7	925B	SS8	92DB	SS9	92C6
SS9A	92C9	SSCHRT	C709	SSET	9258
SST1	9341	SST2	9324	SST3	932D
SST4	9336	SST5	935C	SST6	9365
SSTCOR	930E	SSTCR1	9302	SSTCR2	9317
STAB1	7D27	STAB2	7D51	STADH	D700
STADL	D600	STATAB	8D87	STATB2	8D93
STATUS	0090	STE5	D900	STEST	92EE
STF0	7C8C	STF1	7CB5	STF1.	7CBD
STF2	7CC1	STF3	7C74	STF4	7CC9
STF5	7CB2	STF6	7CD1	STFLG	7C67
STIMME	95F4	STIMMM	95FE	STKPOI	0058
STKSAV	C6AE	STMSK	D800	STRERR	B658
STU	7D20	STUCOL	7D13	STUD8	7D18
STUGR	7D1D	SUNTEN	7F62	SWAIT	9366
SWAIT1	9373	SWAITI	936C	SXH	C63A
SXL	C639	SY	C63B	T1	8028

T1ADR	00A4	T1BUF	96F7	T1CH	96F6
T1CHAD	96FF	T1CP	0061	T1EXIT	9834
T1ILL	9742	T1L10	9814	T1L2	9758
T1L3	9762	T1L3A	976B	T1L4	9782
T1L4A	9793	T1L5	97AD	T1L6	97F2
T1L7	97FD	T1L8	9822	T1L9	981C
T1LEN	9707	T1LP0	9744	T1LP1	97BB
T1LP2	97BD	T1RVS	96F3	T1Y2	96F4
T1YR	96F5	T2	8016	T3	8024
T4	8033	T5	803E	TABTAB	D600
TAU0	8622	TAUSCH	8620	TAZ	7CD4
TCOL	C61D	TES1	9F07	TES2	9F39
TES3	9F3D	TESCOR	7FFF	TESGR	8038
TESHGR	8029	TEST1	9EFD	TEST2	9F0A
TESTE	9E4C	TESTE1	9E73	TESTE2	9E5C
TESTE3	9E62	TEXRAM	0400	TEXT	9708
TEXTM	8B4F	TIMEH	C634	TIMEL	C631
TLPL	7ECB	TNXTCH	9828	TO	00A4
TO0	9535	TO1	9508	TONIRQ	9506
TONTAB	96D8	TPL	82A4	TR1	7D6F
TR2	7DDD	TR4	7DB6	TRA0	82EF
TRA1	82F5	TRA2	8306	TRA3	830C
TRA4	831F	TRA5	833D	TRA6	835C
TRANS	7D5A	TRANSS	82D0	TRCOL	7DD5
TRSB1	7DB4	TRSBEF	7D9C	TSCHAU	9B3F
TTTT	8BA5	TU0	9629	TU1	9634
TU3	9663	TU4	965C	TU5	9673
TU6	96D1	TU7	960C	TU8	9614
TUNE	9602	TZA	7CF7	U.O	8733
UN1	870B	UN2	8724	UN3	8728
UNTEN	86FA	UPL	8244	USE	00FD
V	D000	VARADR	C6AD	VARSTA	002D
VKTNEW	77FC	VKTOLD	0300	VKTSAV	C620
VO1	9545	VOLUM	C674	VOLUME	9538
VRAD1H	0063	VRAD1L	0062	VRAD2H	006C
VRAD2L	006B	W	DD00	WARM1	7C54
WARM2	7C51	WARMST	7C44	WELLFO	C671
WERROR	861D	WINO	85DF	WINOO	8511
WIN1	858A	WIN2	8577	WIN3	857F
WIN4	8551	WIN5	8533	WIN8	85CD
WIN9	85DA	WINIT	8512	WINITO	8593
WINIT2	85E0	WINITS	85A2	WX1H	0058
WX1L	0057	WX2H	005B	WX2L	005A
WXAH	005E	WXAL	005D	WY1	0059
WY2	005C	WYA	005F	X.SREG	C676
X1H	C63F	X1L	C63E	X2H	C643
X2L	C642	X3H	C647	X3L	C646
XK	0014	XXX	955C	Y1	C63C
Y2	C640	Y3	C644	ZA	005F
ZAHL	C686	ZIELAD	86D8	ZLNR	C68C

Im AnschluP an das große Listing der Supergraphik wollen wir Ihnen im folgenden - wie vereinbart - kurz noch die 3 Source-Lisitings der Hardcopyroutinen vorstellen, mit denen die Supergraphik ausgestattet ist:

Zunächst einmal zeigen wir Ihnen hier die Hardcopy-Routine für alle 7-Nadel-Drucker und Epson-Kompatible mit DATA-BECKER-Interface:

```
                ;
                ;HARDCOPY    7-NADELDRUCKER:
                ;****************************
                ;
                ;
                ;
C400 A9 23      HARD.B   LDA  #"#"
C402 86 FD               STX  USE
C404 20 FF AE            JSR  $AEFF    ;PRUEFT AUF ZEICHEN
C407 20 9E B7            JSR  GETBYT
C40A 86 67               STX  $67
C40C 20 0F F3            JSR  $F30F    ;SUCHT LOG. FILENR.
C40F 20 1F F3            JSR  $F31F    ;SETZT FILEPARAM.
C412 A6 67               LDX  $67
C414 20 C9 FF            JSR  $FFC9    ;KANAL OEFFNEN
C417 A9 FF               LDA  #$FF
C419 85 61               STA  $61      ;MASKE
C41B A9 07               LDA  #7
C41D 85 FD               STA  USE      ;PACKENGROESSE
C41F A9 1C               LDA  #28
C421 85 97               STA  FLG      ;ZEILENZAEHLER
C423 A9 00               LDA  #0
C425 8D 48 C6            STA  ZWIS     ;YK-MERKER
C428 A9 28      SEV1     LDA  #40
C42A 85 96               STA  FLG2     ;PACKENZAEHLER
C42C A2 04               LDX  #4
C42E BD C2 C4 SEV1.      LDA  SEVTAB,X  ;MITTENZENTRIERT
C431 20 D2 FF            JSR  $FFD2
C434 CA                  DEX
C435 10 F7               BPL  SEV1.
C437 A9 00               LDA  #0
C439 85 63               STA  $63
C43B 85 64               STA  $64      ;XK=0
C43D AD 48 C6 SEV2       LDA  ZWIS
C440 85 65               STA  $65      ;YK
C442 A9 00               LDA  #0
C444 85 FE               STA  USE+1
C446 A5 01               LDA  1
C448 48                  PHA
C449 29 FC               AND  #$FC
```

```
C44B 78                SEI
C44C 85 01             STA  1
C44E A5 63    SEV3     LDA  $63      ;XK-L
C450 A6 64             LDX  $64      ;XK-H
C452 A4 65             LDY  $65      ;YK
C454 20 C7 C4          JSR  HPOSN    ;POSITION BERECHNEN
C457 A0 00             LDY  #0
C459 B1 AC             LDA  (ADL),Y
C45B A6 FE             LDX  USE+1
C45D 9D 00 C6          STA  $C600,X ;IN BUFFER
C460 E6 65             INC  $65      ;YK
C462 E8                INX
C463 86 FE             STX  USE+1
C465 E4 FD             CPX  USE
C467 D0 E5             BNE  SEV3
C469 68                PLA
C46A 85 01             STA  1
C46C 58                CLI
C46D A9 00             LDA  #0
C46F A0 07             LDY  #7
C471 A6 FD    SEV4     LDX  USE
C473 1E 00 C6 SEV5     ASL  $C600,X
C476 2A                ROL  A
C477 CA                DEX
C478 10 F9             BPL  SEV5
C47A 25 61             AND  $61      ;BEI LETZTER ZEILE=#$0F
C47C 09 80             ORA  #$80     ;HIGH-BYTE
C47E 20 D2 FF          JSR  $FFD2    ;SENDEN
C481 88                DEY
C482 10 ED             BPL  SEV4
C484 A5 63             LDA  $63      ;XK-L
C486 18                CLC
C487 69 08             ADC  #8
C489 85 63             STA  $63      ;XK+8
C48B 90 02             BCC  SEV6
C48D E6 64             INC  $64      ;XK-H
C48F C6 96    SEV6     DEC  FLG2
C491 D0 AA             BNE  SEV2
C493 A9 0D             LDA  #$D
C495 20 D2 FF          JSR  $FFD2
C498 AD 48 C6          LDA  ZWIS
C49B 18                CLC
C49C 69 07             ADC  #7
C49E 8D 48 C6          STA  ZWIS     ;MERKER+7
C4A1 C6 97             DEC  FLG
C4A3 F0 03             BEQ  SEV8.
C4A5 4C 28 C4 SEV8     JMP  SEV1
C4A8 A9 04    SEV8.    LDA  #4
C4AA C5 FD             CMP  USE
C4AC F0 0C             BEQ  SEV7
C4AE 85 FD             STA  USE      ;LETZTE ZEILE
```

```
C4B0 A9 01           LDA  #1
C4B2 85 97           STA  FLG
C4B4 A9 0F           LDA  #$F
C4B6 85 61           STA  $61     ;MASKE
C4B8 D0 EB           BNE  SEV8
C4BA A9 0F    SEV7   LDA  #15     ;NORMAL MODUS
C4BC 20 D2 FF        JSR  $FFD2
C4BF 4C CC FF        JMP  $FFCC   ;SCHLIESST KANAL
              ;
C4C2 50 00 10 SEVTAB .BYT 80,0,16,27,8
              ;
C4C7 85 14    HPOSN  STA  XK      ;S. HAUPTPROGRAMM
C4C9 86 15           STX  XK+1
C4CB 98              TYA
C4CC 4A              LSR  A
C4CD 4A              LSR  A
C4CE 4A              LSR  A
C4CF AA              TAX
C4D0 BD 01 C5        LDA  MUL.H,X
C4D3 85 AD           STA  ADH
C4D5 8A              TXA
C4D6 29 03           AND  #3
C4D8 AA              TAX
C4D9 BD 1A C5        LDA  MUL.L,X
C4DC 85 AC           STA  ADL
C4DE 98              TYA
C4DF 29 07           AND  #7
C4E1 18              CLC
C4E2 65 AC           ADC  ADL
C4E4 85 AC           STA  ADL
C4E6 A5 14           LDA  XK
C4E8 29 F8           AND  #$F8
C4EA 85 63           STA  OFFX
C4EC AD 4F C6        LDA  GRMEM   ;LAUFENDE GRAPHIKSEITE
C4EF 05 AD           ORA  ADH
C4F1 85 AD           STA  ADH
C4F3 18              CLC
C4F4 A5 AC           LDA  ADL
C4F6 65 63           ADC  OFFX
C4F8 85 AC           STA  ADL
C4FA A5 AD           LDA  ADH
C4FC 65 15           ADC  XK+1
C4FE 85 AD           STA  ADH
C500 60              RTS
                     ;
C501 00 01 02 MUL.H  .BYT 0,1,2,3,5,6,7,8,10,11,12,13,15,16
C50F 11 12 14        .BYT 17,18,20,21,22,23,25,26,27,28,30
C51A 00 40 80 MUL.L  .BYT 0,$40,$80,$C0
              ;
```

Die folgende Routine für die beiden Drucker CBM 1526 und MPS 802 nutzt die Möglichkeit der Drucker, ein einziges Sonderzeichen zu definieren. Nach der Ausgabe dieses Sonderzeichens muP zunächst wieder an den Zeilenanfang und dann zur aktuellen Position gesprungen werden. Dies erhöht entsprechend den Programmieraufwand:

```
                ;
                ;HARDCOPY   CBM 1526:
                ;*********************
                ;
                ;
0068            HHZF     =    $68
                ;
C400 A9 23      HARD.A   LDA  #"#"
C402 86 FD               STX  USE
C404 20 FF AE            JSR  $AEFF   ;PRUEFT AUF ZEICHEN
C407 20 9E B7            JSR  GETBYT
C40A 86 67               STX  $67
C40C 20 0F F3            JSR  $F30F   ;SUCHT LOG. FILENR.
C40F 20 1F F3            JSR  $F31F   ;SETZT FILEPARAM.
C412 A9 05      EIGHT    LDA  #5
C414 20 39 C5            JSR  OPENCH  ; OPEN FUER SA=5
C417 A9 06               LDA  #6
C419 20 39 C5            JSR  OPENCH  ; OPEN FUER SA=6
C41C A2 16               LDX  #$16
C41E 20 C9 FF            JSR  $FFC9
C421 A9 14               LDA  #20     ; ZEILENABSTAND FUER GRAFIK
C423 20 D2 FF            JSR  $FFD2
C426 A9 0D               LDA  #13     ; ENDEZEICHEN
C428 20 D2 FF            JSR  $FFD2
C42B 20 CC FF            JSR  $FFCC   ; KANAL LOESCHEN
C42E A0 19               LDY  #25     ; ANZAHL ZEILEN
C430 84 64               STY  $64     ; SETZEN
C432 A9 02      HA10     LDA  #2      ; DOPPELT HOCH
C434 85 68               STA  HHZF    ; WIEDERHOLZAEHLER
C436 A9 00      HA10A    LDA  #0      ; ZAEHLER FUER FUEHRENDE BLANKS
C438 85 65               STA  $65     ; SETZEN
C43A A9 28               LDA  #40     ; ANZAHL SPALTEN
C43C 85 63               STA  $63     ; SETZEN
C43E A9 19               LDA  #25
C440 38                  SEC
C441 E5 64               SBC  $64     ;ZEILEN NOCH ZU DRUCKEN
C443 AA                  TAX
C444 BD 42 C5            LDA  MUL.H,X
C447 0D 4F C6            ORA  GRMEM
C44A 85 AD               STA  ADH
C44C 8A                  TXA
C44D 29 03               AND  #3
C44F AA                  TAX
```

```
C450 BD 5C C5           LDA  MUL.L,X
C453 85 AC              STA  ADL
C455 A2 00    HA11      LDX  #0        ; GRAFIKDATEN HOLEN
C457 A0 08              LDY  #8
C459 78                 SEI
C45A A5 01              LDA  1
C45C 48                 PHA
C45D 29 FC              AND  #$FC
C45F 85 01              STA  1
C461 A1 AC    HA1L2     LDA  (ADL,X)
C463 99 00 C6           STA  $C600,Y
C466 E6 AC              INC  ADL
C468 D0 02              BNE  HA1L01
C46A E6 AD              INC  ADH
C46C 88       HA1L01    DEY
C46D D0 F2              BNE  HA1L2
C46F 68                 PLA
C470 85 01              STA  1
C472 58                 CLI
C473 A0 08              LDY  #8
C475 A2 08    HA1L4     LDX  #8
C477 1E 00 C6 HA1L3     ASL  $C600,X
C47A 2A                 ROL  A
C47B CA                 DEX
C47C D0 F9              BNE  HA1L3
C47E 20 18 C5           JSR  BSPLIT
C481 99 08 C6           STA  $C608,Y
C484 88                 DEY
C485 D0 EE              BNE  HA1L4
C487 A0 08    HA1L02    LDY  #8        ; GRAFIK DRUCKEN
C489 20 BA C4           JSR  PRDFCH
C48C A0 04              LDY  #4
C48E 20 BA C4           JSR  PRDFCH
C491 C6 63              DEC  $63
C493 D0 C0              BNE  HA11
C495 C6 68              DEC  HHZF
C497 D0 9D              BNE  HA10A
C499 C6 64    HA1L50    DEC  $64       ; ZEILENZAEHLER ERNIEDRIGEN
C49B D0 95              BNE  HA10      ; NOCH NICHT FERTIG
C49D A2 16              LDX  #$16
C49F 20 C9 FF           JSR  $FFC9     ; SA=6
C4A2 A9 24              LDA  #36
C4A4 20 D2 FF           JSR  $FFD2     ; ZEILENABSTAND TEXT
C4A7 A9 0D              LDA  #13
C4A9 20 D2 FF           JSR  $FFD2     ; ENDEZEICHEN
C4AC 20 CC FF           JSR  $FFCC
C4AF A9 16              LDA  #$16      ; SA=6
C4B1 20 C3 FF           JSR  $FFC3
C4B4 A9 19              LDA  #25
C4B6 20 C3 FF           JSR  $FFC3     ; CLOSE FUER SONDERZEICHENKANAL
C4B9 60                 RTS            ; FERTIG
```

```
C4BA A9 00    PRDFCH  LDA  #0
C4BC 85 66            STA  $66
C4BE A2 15            LDX  #$15
C4C0 20 C9 FF         JSR  $FFC9    ; KANAL FUER ZEICHENDEFINITION
C4C3 98               TYA
C4C4 48               PHA
C4C5 A2 04            LDX  #4
C4C7 B9 08 C6 HA62    LDA  $C608,Y
C4CA 20 D2 FF         JSR  $FFD2
C4CD 20 D2 FF         JSR  $FFD2
C4D0 05 66            ORA  $66
C4D2 85 66            STA  $66
C4D4 88               DEY
C4D5 CA               DEX
C4D6 D0 EF            BNE  HA62
C4D8 20 CC FF         JSR  $FFCC    ; KANAL SCHLIESSEN
C4DB A5 66            LDA  $66
C4DD F0 31            BEQ  HA61
C4DF A6 67            LDX  $67
C4E1 20 C9 FF         JSR  $FFC9    ; DATENKANAL
C4E4 A5 65            LDA  $65      ; ANZAHL BLANKS
C4E6 F0 09            BEQ  HA63     ; KEINE
C4E8 AA               TAX
C4E9 A9 20            LDA  #$20
C4EB 20 D2 FF HA64    JSR  $FFD2
C4EE CA               DEX
C4EF D0 FA            BNE  HA64
C4F1 A9 FE    HA63    LDA  #254     ; SONDERZEICHEN
C4F3 20 D2 FF         JSR  $FFD2
C4F6 68       HA61A   PLA
C4F7 C9 04            CMP  #4
C4F9 D0 0A            BNE  HA75A
C4FB A5 63            LDA  $63
C4FD C9 01            CMP  #1
C4FF D0 04            BNE  HA75A
C501 A9 0D            LDA  #13
C503 D0 02            BNE  HA75
C505 A9 8D    HA75A   LDA  #141
C507 20 D2 FF HA75    JSR  $FFD2
C50A 20 CC FF         JSR  $FFCC    ; KANAL SCHLIESSEN
C50D E6 65            INC  $65
C50F 60               RTS
C510 A6 67    HA61    LDX  $67      ;LEERDEFINITION
C512 20 C9 FF         JSR  $FFC9
C515 4C F6 C4         JMP  HA61A
              ;
C518 A6 68    BSPLIT  LDX  HHZF     ; ZEILENFLAG
C51A E0 01            CPX  #1       ; DRUCK DER ZWEITEN ZEILE PRINT
C51C F0 04            BEQ  HA90     ; JA
C51E 4A               LSR  A
C51F 4A               LSR  A
```

```
C520 4A               LSR  A
C521 4A               LSR  A         ; OBERES HALBBYTE NACH UNTEN
C522 29 0F     HA90   AND  #$0F      ; TABELLENINDEX BILDEN
C524 AA               TAX
C525 BD 29 C5         LDA  HATAB1,X  ; SPLITWERT HOLEN ]
C528 60               RTS
               ;
C529 00        HATAB1 .BYT %00000000
C52A 03               .BYT %00000011
C52B 0C               .BYT %00001100
C52C 0F               .BYT %00001111
C52D 30               .BYT %00110000
C52E 33               .BYT %00110011
C52F 3C               .BYT %00111100
C530 3F               .BYT %00111111
C531 C0               .BYT %11000000
C532 C3               .BYT %11000011
C533 CC               .BYT %11001100
C534 CF               .BYT %11001111
C535 F0               .BYT %11110000
C536 F3               .BYT %11110011
C537 FC               .BYT %11111100
C538 FF               .BYT %11111111
               ;
C539 85 B9     OPENCH STA  $B9
C53B 09 10            ORA  #$10
C53D 85 B8            STA  $B8
C53F 4C C0 FF         JMP  $FFC0
               ;
C542 00 01 02 MUL.H   .BYT 0,1,2,3,5,6,7,8,10,11,12,13,15,16
C550 11 12 14         .BYT 17,18,20,21,22,23,25,26,27,28,30,31
C55C 00 40 80 MUL.L   .BYT 0,$40,$80,$C0
               ;
```

Die letzte Hardcopy-Routine bringt ein farbiges Bild mit dem Drucker Seikosha GP 700A zu Papier:

```
                ;
                ;HARDCOPY":FARBDRUCKER
                ;********************
                ;
C400                      *=   $C400
                ;
C400 C9 44                CMP  #"D"    ;DEFINITIONPRINT
C402 D0 18                BNE  FARBDR
C404 20 73 00             JSR  CHRGET  ;FARBANPASSUNG
C407 20 9E B7             JSR  GETBYT
C40A 8A                   TXA
C40B 29 0F                AND  #$F
C40D 48                   PHA          ;BILDSCHIRMFARBE
C40E 20 F1 B7             JSR  CHKGET
C411 8A                   TXA
C412 29 07                AND  #7
C414 A8                   TAY          ;DRUCKFARBE
C415 68                   PLA
C416 AA                   TAX
C417 98                   TYA
C418 9D 99 C6             STA  DCOLTB,X  ;BILDSCHIRMFARBE ZUORDNEN
C41B 60                   RTS
C41C 20 73 00 FARBDR      JSR  CHRGET
C41F A9 23                LDA  #"#"
C421 86 FD                STX  USE
C423 20 FF AE             JSR  $AEFF   ;PRUEFT AUF ZEICHEN
C426 20 9E B7             JSR  GETBYT
C429 86 67                STX  $67
C42B 20 0F F3             JSR  $F30F   ;SUCHT LOG. FILENR.
C42E 20 1F F3             JSR  $F31F   ;SETZT FILEPARAM.
C431 A6 67                LDX  $67     ;LOG. FILENUMMER
C433 20 C9 FF             JSR  $FFC9   ;KANAL OEFFNEN
C436 20 F1 B7             JSR  CHKGET
C439 CA                   DEX          ;1 ODER 2 -PUNKTGRAPHIK
C43A 8A                   TXA
C43B A0 0F                LDY  #15
C43D 4A                   LSR  A
C43E B0 01                BCS  FD0
C440 88                   DEY          ;1PUNKT
C441 2A       FD0         ROL  A
C442 AA                   TAX
C443 E8                   INX
C444 86 63                STX  $63
C446 86 97                STX  FLG     ;DOPPLUNGSZAEHLER
C448 B9 44 C5 FD1         LDA  DRCKTB,Y
C44B 20 D2 FF             JSR  $FFD2   ;SENDEN":ESC P 640400
C44E 88                   DEY          ;BZW.   ESC P 480200
```

```
C44F 88               DEY
C450 10 F6            BPL  FD1
C452 A2 00            LDX  #0
C454 8E 48 C6         STX  ZWIS     ;ZEILENZAEHLER
C457 BD 54 C5 FD2     LDA  MUL.H,X
C45A 0D 4F C6         ORA  GRMEM
C45D 85 AD            STA  ADH
C45F 8A               TXA
C460 29 03            AND  #3
C462 AA               TAX
C463 BD 6D C5         LDA  MUL.L,X
C466 85 AC            STA  ADL      ;ADRESSE ERRECHNEN
C468 A9 08            LDA  #8
C46A 85 FD            STA  USE      ;REIHENZAEHLER
C46C A9 28     FD4    LDA  #40
C46E 85 AB            STA  MSK      ;REIHENBYTEZAEHLER
C470 A6 63            LDX  $63
C472 CA               DEX
C473 D0 0A            BNE  FD5
C475 A2 A0            LDX  #160     ;1-PUNKTGRAPHIK
C477 A9 00            LDA  #0       ;=> IN DIE MITTE SCHIEBEN
C479 20 D2 FF FD3     JSR  $FFD2
C47C CA               DEX
C47D D0 FA            BNE  FD3
C47F A5 AC     FD5    LDA  ADL
C481 85 61            STA  $61
C483 A5 AD            LDA  ADH
C485 4A               LSR  A
C486 66 61            ROR  $61
C488 4A               LSR  A
C489 66 61            ROR  $61
C48B 4A               LSR  A
C48C 66 61            ROR  $61      ;/8
C48E 49 DC            EOR  #%11011100  ;JEWEILIGES VIDEORAM
C490 85 62            STA  $62      ;FARBADRESSE
C492 A0 00            LDY  #0
C494 B1 61            LDA  ($61),Y ;HOLE WERT
C496 48               PHA
C497 29 0F            AND  #$F      ;HGRUND IN HGR/COL1 IN MC
C499 AA               TAX
C49A 68               PLA
C49B 4A               LSR  A
C49C 4A               LSR  A
C49D 4A               LSR  A
C49E 4A               LSR  A        ;COL1 IN HGR/COL2 IN MC
C49F 8D 01 C6         STA  $C601    ;COL1
C4A2 24 02            BIT  GFLAG
C4A4 10 18            BPL  FD6      ;HGR
C4A6 8E 02 C6         STX  $C602    ;COL2 / ACHTUNG! UNSTIMMIGKEIT
C4A9 A5 62            LDA  $62      ;COL1+2 NACH HOLZHACKERMETHODE
C4AB 29 03            AND  #3       ;BEHOBEN!
```

```
C4AD 09 D8             ORA  #$D8     ;FARBRAM
C4AF 85 62             STA  $62
C4B1 B1 61             LDA  ($61),Y ;COL3
C4B3 29 0F             AND  #$F
C4B5 8D 03 C6          STA  $C603
C4B8 AD 21 D0          LDA  V+33     ;HGRUND
C4BB 29 0F             AND  #$F
C4BD AA                TAX
C4BE 8E 00 C6 FD6      STX  $C600    ;HGRUND
C4C1 A5 01             LDA  1
C4C3 AA                TAX
C4C4 29 FC             AND  #$FC
C4C6 78                SEI
C4C7 85 01             STA  1
C4C9 A9 08             LDA  #8
C4CB 85 96             STA  FLG2     ;BITZAEHLER
C4CD B1 AC             LDA  (ADL),Y ;HOLE BYTE
C4CF 48                PHA
C4D0 8A                TXA
C4D1 85 01             STA  1
C4D3 58                CLI
C4D4 A9 00    FD7      LDA  #0
C4D6 85 FE             STA  USE+1
C4D8 68                PLA
C4D9 24 02             BIT  GFLAG
C4DB 10 05             BPL  FD8      ;HGR
C4DD 0A                ASL  A
C4DE 26 FE             ROL  USE+1
C4E0 C6 96             DEC  FLG2     ;BITZAEHLER
C4E2 0A       FD8      ASL  A
C4E3 26 FE             ROL  USE+1
C4E5 48                PHA             ;REST MERKEN
C4E6 A6 FE             LDX  USE+1      ;FARBNUMMER
C4E8 BD 00 C6          LDA  $C600,X    ;BILDSCHIRMFARBE
C4EB AA                TAX
C4EC BD 99 C6          LDA  DCOLTB,X   ;ZUGEORDNETE DRUCKFARBE
C4EF 48                PHA
C4F0 A5 63             LDA  $63
C4F2 24 02             BIT  GFLAG
C4F4 10 01             BPL  FD9.
C4F6 0A                ASL  A
C4F7 AA       FD9.     TAX
C4F8 68                PLA
C4F9 20 D2 FF FD9      JSR  $FFD2
C4FC CA                DEX
C4FD D0 FA             BNE  FD9      ;2*2-PUNKTE/MC":2PUNKTE
C4FF C6 96             DEC  FLG2     ;BITZAEHLER
C501 D0 D1             BNE  FD7
C503 68                PLA
C504 A5 AC             LDA  ADL
C506 18                CLC
```

```
C507 69 08              ADC  #8
C509 85 AC              STA  ADL
C50B 90 02              BCC  FD10
C50D E6 AD              INC  ADH      ;ADRESSE+8
C50F C6 AB     FD10     DEC  MSK      ;BYTEZAEHLER
C511 F0 03              BEQ  FD11
C513 4C 7F C4           JMP  FD5      ;NAECHSTES BYTE
C516 18        FD11     CLC
C517 C6 97              DEC  FLG      ;DOPPLUNGSZAEHLER
C519 D0 09              BNE  FD12
C51B A5 63              LDA  $63
C51D 85 97              STA  FLG
C51F C6 FD              DEC  USE      ;REIHENZAEHLER
C521 F0 10              BEQ  FD13
C523 38                 SEC
C524 A5 AC     FD12     LDA  ADL
C526 E9 3F              SBC  #<319
C528 85 AC              STA  ADL
C52A A5 AD              LDA  ADH
C52C E9 01              SBC  #>319
C52E 85 AD              STA  ADH      ;319/320 ABZIEHEN
C530 4C 6C C4           JMP  FD4      ;NAECHSTE REIHE
C533 AE 48 C6 FD13      LDX  ZWIS     ;ZEILENZAEHLER
C536 E8                 INX
C537 8E 48 C6           STX  ZWIS
C53A E0 19              CPX  #25
C53C F0 03              BEQ  FD14
C53E 4C 57 C4           JMP  FD2      ;NAECHSTE ZEILE
               ;
C541 4C CC FF FD14      JMP  $FFCC    ;KANAL SCHLIESSEN
               ;
               ;
               ;1PUNKTGR": ESC P 480200
               ;2PUNKTGR": ESC P 640400
               ;
C544 30 30 30 DRCKTB    .ASC "000024008446PP"
C552 1B 1B              .BYT 27,27
C554 00 01 02 MUL.H     .BYT 0,1,2,3,5,6,7,8,10,11,12,13,15,16
C562 11 12 14           .BYT 17,18,20,21,22,23,25,26,27,28,30
C56D 00 40 80 MUL.L     .BYT 0,$40,$80,$C0
               ;
```

7. Anhang

7.1 Bits and Bytes - Kleine Computermathematik

Um die vielen verschiedenen Faktoren festzulegen, mit denen Sie die zahllosen Möglichkeiten der Graphikvariation bestimmen können, werkeln Sie direkt in der Speicherstruktur Ihres Gerätes bzw. mit den unterschiedlichen Registern (s.u.) der integrierten Schaltkreise. Bekommen Sie keinen Schreck! Wir wollen Sie nicht etwa mit elektronischen Einzelheiten bombardieren. Zum Verständnis dieser Dinge bedarf es nur etwas Mathematik, die relativ einfach zu durchschauen ist und oft schon zum Stoff von 5- oder 6-Klässlern gehört. Es geht unter anderem um die Darstellung von Zahlen im sogenannten Binär- oder Dualsystem.

Für einen gewöhnlichen Computer, der ja bekanntlich aus einer Unzahl von elektronischen Leitungen und Bausteinen besteht, gibt es lediglich zwei Zustände, aus denen seine ganze kleine Welt besteht: Strom ein - Strom aus. Da wir Menschen nun aber von ihm eine ganze Menge mehr verlangen, mußten wir uns etwas einfallen lassen, um mit diesem Mangel zu leben. Wollen wir zum Beispiel eine Zahl im Computer darstellen, so kommen wir nicht mit diesen beiden Zuständen aus. Wir könnten zwar dem Zustand "Strom aus" die Zahl 0 und "Strom ein" die Zahl 1 zuordnen, werden damit aber wohl nicht weit kommen, da wir außer 0 und 1 noch unendlich viele andere Zahlen darstellen wollen.

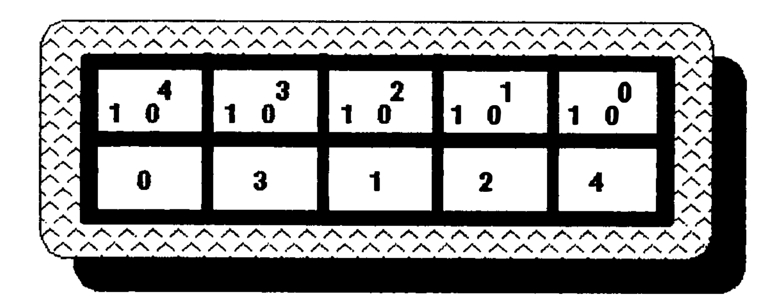
7.1.1 Dezimalsystem

Im alltäglichen Leben stehen uns 10 Ziffern (0-9) zur Verfügung (deshalb der Name "Dezimal"-System von decem lat. - zehn), mit denen wir durch einen kleinen Trick sämtliche weiteren Zahlen darstellen können: Wir reihen einfach mehrere Ziffern zu einer großen Zahl zusammen. Wollen wir beispielsweise bis 1000 zählen, so sind wir bereits bei der Ziffer 9 am Ende unseres Ziffernvorrates. Jedem ist bekannt, daß wir danach einfach wieder von vorne (bei 0) anfangen zu zählen, wobei wir jedoch als Kennzeichen, daß wir bereits einmal bei 9 angelangt sind eine 1 vor die laufende Ziffer schreiben. Die nächsten Zahlen nach 9 lauten bekanntlich 10 (eins null), 11 (eins eins), 12 (eins zwei) usw. Bei 19 sehen wir uns dem gleichen Problem gegenübergestellt. Doch wieder beginnen wir bei 0 und erhöhen lediglich die vorgestellte Ziffer um eins. Das Ergebnis: 20 (zwei null). Sind

wir bei dieser ersten Ziffer ebenfalls bei 9 angekommen (99), so beginnen wir hier gleichfalls einfach wieder bei 0 und stellen davor die Zahl 1 (100). Nach diesem Prinzip werden schließlich sämtliche ganzen Zahlen dargestellt. Dabei werden die einzelnen Ziffern einer Zahl Stellen genannt und diese wiederum als Einer, Zehner, Hunderter, Tausender, ... bezeichnet. Eine Zahl z, deren Ziffern (d^1, d^2,...) bekannt sind, läßt sich somit wie folgt errechnen:

$$z = d_0*10^0 + d_1*10^1 + d_2*10^2 + \ldots$$

wobei d jeweils die Einer-, Zehner-, Hunderter-, ... -stellen bezeichnet (eine kleine Anmerkung: eine Zahl hoch 0 ergibt stets 1, also 10 hoch 0 ist gleich 1, genauso wie z.B. 5 hoch 0 gleich eins ist - Ausnahme: 0 hoch 0 ist nicht definiert). Eine andere Darstellung ist die folgende (hier an dem Beispiel der willkürlich gewählten Zahl 3124):



Die obere Zeile nennt dabei den Wert der einzelnen Stellen, die untere Zeile den Faktor d.

7.1.2 Dualsystem

Wir besitzen, wie gesagt, zehn Ziffern, um unsere Zahlen einigermaßen übersichtlich darzustellen. Unser armer Computer muß sich jedoch mit 2 Ziffern begnügen (0 und 1), wie oben dargelegt. Wie aber zählt er dann bis 1000? Ganz einfach: genauso wie wir! Also:

0, 1

damit ist sein Ziffernschatz "verbraucht" und er startet wieder bei 0, setzt aber gleichfalls als Kennzeichen eine 1 davor und erhält:

10

Es geht weiter mit:

11,100,101,110,111,1000,1001,1010,1011,1100,1101,....

Wie Sie sehen, ist das sogenannte Binärsystem völlig analog zu dem uns vertrauten Dezimalsystem aufgebaut. Entsprechend läßt sich der Wert einer Dualzahl durch die folgende Formel errechnen:

$z = b_0*2^0 + b_1*2^1 + b_2*2^2 + ...$

wobei die Parameter b_1, b_0, b_2, ... die einzelnen Ziffern, angefangen von der ersten (Einerstelle) bis zur höchsten Dualstelle, darstellen. Kennen Sie also die Ziffern einer Dualzahl, so ist es Ihnen mit Hilfe dieser Formel möglich, sie in eine Dezimalzahl umzurechnen. Wie oben kann dies ebenfalls durch die folgende Tabelle erreicht werden (hier an dem Beispiel der Zahl 10110100):

$2^7 =$	$2^6 =$	$2^5 =$	$2^4 =$	$2^3 =$	$2^2 =$	$2^1 =$	$2^0 =$
128	64	32	16	8	4	2	1
1	0	1	1	0	1	0	0
128		+ 32	+ 16		+ 4		

Eine solche Stelle nennt man nun in der Computerfachsprache ein Bit, d.h. eine Informationseinheit. Unter Informationseinheit (oder Bit) versteht man also die Möglichkeit zweier Zustände (ja oder nein bzw. 1 oder 0). In Ihrem Rechner sind nun 8 solcher Bits (oder Dualstellen) zu einer Einheit zusammengefaßt, dem sogenannten Byte. Mit diesen 8 Bits lassen sich also Zahlen von 0-255 darstellen. Mit zwei Byte (=16 Bits) dagegen schon von 0-65535. Wollen wir ein Byte also in eine Dezimalzahl umrechnen (was notwendig wird, wenn wir von Basic aus Änderungen direkt an bestimmten Speicherstellen vornehmen wollen), so verwenden wir selbstverständlich wieder unsere Tabelle (oder die Formel).

Wollen wir dagegen umgekehrt eine Dezimalzahl in eine Dualzahl umrechnen, so gehen wir wie folgt vor (am Beispiel der Zahl 180):

```
180 | 128 | = | 1 Rest 52
 52 |  64 | = | 0 Rest 52
 52 |  32 | = | 1 Rest 20
 20 |  16 | = | 1 Rest  4
  4 |   8 | = | 0 Rest  4
  4 |   4 | = | 1 Rest  0
  0 |   2 | = | 0 Rest  0
  0 |   1 | = | 0 Rest  0
```

$180_{(d)} = 10110100_{(b)}$

Hier wurde also die umzurechnende Zahl 180 nacheinander durch die Potenzwerte aus der Tabelle geteilt. Das Ergebnis stellt dabei jeweils eine Ziffer der gewünschten Dualzahl dar, der Rest dieser Division wird für die übrigen Rechnungen weiterverwendet.

Eine zweite Methode ist die folgende. Sie eignet sich besonders für Computerprogramme, da sie rekursiv und damit besonders einfach und schnell ist:

```
180 | 2 = 90 Rest 0
 90 | 2 = 45 Rest 0
 45 | 2 = 22 Rest 1
 22 | 2 = 11 Rest 0
 11 | 2 =  5 Rest 1
  5 | 2 =  2 Rest 1
  2 | 2 =  1 Rest 0
  1 | 2 =  0 Rest 1
```

Hier wird, wie Sie sehen, ständig durch 2 geteilt. Der jeweilige Rest stellt dabei startend von der Einerstelle eine Dualziffer dar. Das Ergebnis der Divison wird weiter für die folgenden Rechnungen verwandt, solange, bis es 0 wird.

Um zu kennzeichnen, daß es sich bei einer bestimmten Zahl um eine Dualzahl handelt, setzen wir vereinbarungsgemäß ein Prozentzeichen (%) vor die jeweilige Zahl. Dies ist üblich und wird auch im weiteren verwendet.

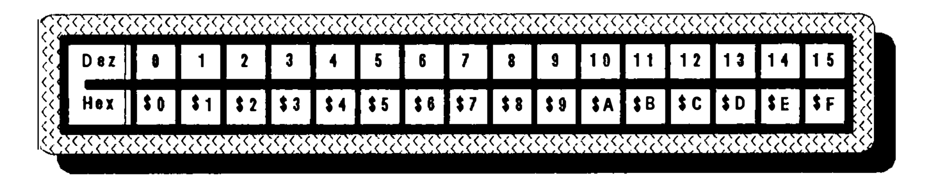
Wie Sie in den ganzen Ausführungen sehen, werden unsere Zahlen auf die Dauer umständlich lang und unübersichtlich. Der Computer kann damit sehr viel besser umgehen. Wir aber sehnen uns nach unserer guten alten Dezimalschreibweise. Doch hier ist die Umrechnung stets etwas schwierig, wie Sie sahen. Aus diesem Grunde hat man sich etwas anderes ausgedacht, das:

7.1.3 Hexadezimalsystem

Das Hexadezimal- oder 16er-System bietet hier einige Vorteile, die Sie im folgenden kennenlernen werden. Es besitzt 16 verschiedene Ziffern. Da wir von unserem Dezimalsystem her jedoch nur 10 Ziffern kennen, müssen wir sechs weitere erfinden. Alle verfügbaren Ziffern lauten:

Dez	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F

Die fehlenden Ziffern wurden also durch Buchstaben symbolisiert. Um eine Hexadezimalzahl als solche zu kennzeichen, ist es üblich, ein $ (Dollarzeichen) vor diese Zahl zu setzen. Wie Sie sich denken können, läuft die Berechnung oder Umrechnung zwischen den einzelnen Systemen völlig analog. Mithilfe der folgenden Tabelle rechnen Sie eine Hexadezimalzahl in eine 10er-Zahl um (am Beispiel $FE2A):



Dez	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F

Diese Verwandlung ist ebenfalls rekursiv möglich.

Die umgekehrte Rechnung lautet dagegen:

```
65066 | 4096 = | 15 | Rest 3626
 3626 |  256 = | 14 | Rest   42
   42 |   16 = |  2 | Rest   10
   10 |    1 = | 10 | Rest    0
-------------------------------
    65066(d)  =   FE2A(h)
```

Sie sehen, diese Umrechnung folgt den selben Gesetzmäßigkeiten, wie unter Dualzahlen beschrieben. Die dort erwähnte rekursive Methode ist hier selbstverständlich ebenfalls anwendbar.

Welchen Vorteil bietet nun das beschriebene Hexadezimalsystem? Zunächst einmal ist es durch dieses System möglich, Zahlen sehr kurz und übersichtlich anzugeben. Dies allein ist jedoch noch nicht ausschlaggebend. Wichtig ist, daß sich die Umrechnung von Hexadezimal- in Dualzahlen äußerst einfach gestaltet. Jeweils 4 Binärziffern nämlich ergeben stets eine Hexadezimalziffer. Ein Byte kann also durch 2 Hexziffern festgelegt werden, z.B.:

```
% 1101 0110
$   D    6
```

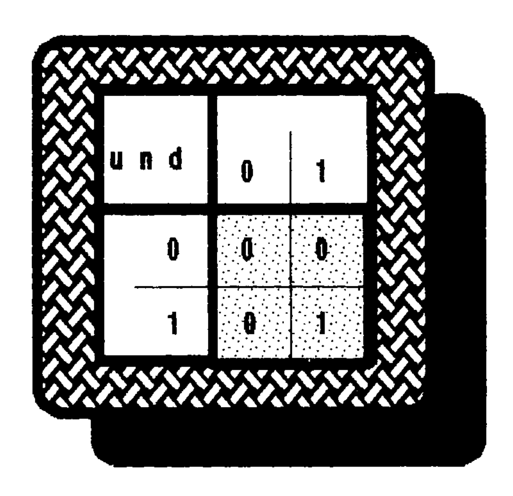
Sie sehen, wie einfach und übersichtlich so eine sonst umständlich lange Dualzahl beschrieben werden kann. Hier im Anhang finden Sie zu Ihrer Unterstützung Dez-Hex-Dual-Konversionstabellen.

7.1.4 Logische Operationen

Um später die Inhalte bestimmter Bytes oder Speicherstellen zu verändern oder zu manipulieren, sind verschiedene logische Operationen nützlich, die auch vom Commodore-Basic aus angewählt werden können und von denen hier drei kurz beschrieben werden sollen: AND, OR und EOR. Bei allen dreien sind stets zwei miteinander zu verknüpfende Dualzahlen notwendig.

a) AND:

Nehmen wir einmal an, Sie wollen ein bestimmtes Bit in einem Byte nur dann erhalten, wenn es gleichzeitig in einem anderen Byte steht. In diesem Falle verwenden Sie die AND-Verknüpfung. Sie wird Bit für Bit vorgenommen und kann durch die folgende Verknüpfungstabelle beschrieben werden:



und	0	1
0	0	0
1	0	1

In der obersten und in der äußerst linken Reihe stehen jeweils die beiden zu verknüpfenden Werte. Im Schnittpunkt der jeweiligen Zeilen und Spalten finden Sie dann das Ergebnis der logischen Verknüpfung. Sie sehen also, daß das Ergebnis nur dann gleich 1 ist, wenn beide verknüpften Bits ebenfalls gleich 1 sind, andernfalls bleibt alles 0. Wollen wir nun zwei vollständige Bytes (bestehend aus je 8 Bits) miteinander "ANDieren", so sieht das Ganze so aus:

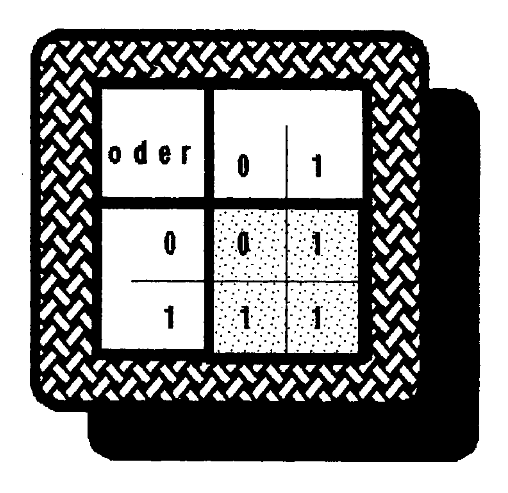
```
    10110010
AND 01100111
------------
    00100010
```

Hier wurde also jedes einzelne Bit des ersten mit dem korrespondierenden Bit des zweiten Bytes durch AND miteinander verknüpft.

Diese Operation wird neben der als nächstes beschriebenen ständig verwendet, um z.B. lediglich ein Bit eines Bytes zu verändern, während die anderen erhalten bleiben.

b) OR:

Die OR-Verknüpfung kann - wie auch AND - durch eine sogenannte Verknüpfungstabelle dargestellt werden:



Wie ersichtlich wird das Ergebis schon dann 1, wenn bereits eines der beiden zu verknüpfenden Bits gleich 1 ist. Ein Byte kann somit etwa so geORt werden:

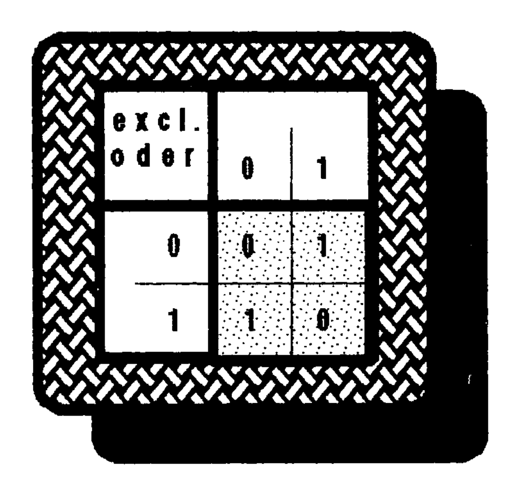
```
   10110010
OR 01100111
   11110111
```

b) EOR:

Die EOR oder exklusiv-ODER-Verknüpfung wird wie folgt in einer Verknüpfungstabelle dargestellt:

Bei dieser Verknüpfung wird nur dann ein Ergebnisbit gleich 1, wenn beide Ausgangsbits unterschiedlich sind. Haben Sie beide den gleichen Wert (beide 1 oder beide 0), dann wird das Ergebnis 0. Die exklusiv-ODER-Verknüpfung eines Bytes sieht dann wie folgt aus:

```
    10110010
EOR 11110111
    01000101
```



7.2 Anhang zur allgemeinen Graphik

7.2.1 Programmoptimierung

In den einzelnen Kapiteln haben wir Ihnen viele Basicprogramme vorgestellt. Doch stört uns oft Ihre Langsamkeit, was nicht selten schöne Effekte verschleiert. Im folgenden werden Ihnen einige Tips gegeben, wie Sie Ihre Basicprogramme friesieren können.

Man unterscheidet grundsätzlich zwei Methoden der Geschwindigkeits-Aufbesserung:

1.) Das Optimieren des Basicprogrammes selbst
2.) Das Ersetzen von langwierigen Basicroutinen durch entsprechende Assemblerprogramme.

Zum ersten Punkt seien hier einige geraffte Informationen gegeben:

- Vermeiden Sie REM-Zeilen (zumindest oder besonders in oft zu durchlaufenden Schleifen).

- Vermeiden Sie zu viele Zeilen. Oft zu durchlaufende Schleifen etc. sollten in möglichst wenigen Zeilen untergebracht werden (nutzen Sie die Befehlsabkürzungen, um möglichst viele Befehle in eine Zeile zu bekommen).

- Vermeiden Sie Leerzeichen zwischen oder in den Befehlen, die nicht syntax- oder programmnotwendig sind.

- Verringern Sie den Rechenaufwand in zeitkritischen Schleifen, indem Sie wenn dies möglich ist, Rechnungen oder Teilrechnungen bereits vor Aufruf der Schleife durchführen und die Ergebnisse in Zwischenspeichern bereithalten.

- Vermeiden Sie das direkte Rechnen mit Zahlen (Konstanten) in Schleifen; besser ist, wenn Sie diese vor Aufruf der Schleife in entsprechende Zischenspeicher packen, da Zahlen immer erst in das rechnerinterne Floatingpoint-Format umgerechnet werden müssen.

- Vermeiden Sie Unterprogrammaufrufe (GOSUBs) oder GOTOs in zeitrelevanten Schleifen.

- Konstruieren Sie Schleifen möglichst nur mit FOR...NEXT - Vermeiden Sie IF.

- Definieren Sie viel gebrauchte (vor allem in Schleifen verwendete) Speicher möglichst zuerst. Es genügt z.B. ein X=0 am Programmanfang, wenn Sie diese Variable im Laufe des Programmes sehr häufig verwenden.

- Packen Sie zusammengehörige Programmteile möglichst nahe beieinander, um langes Suchen des Rechners nach der Zeilennummer bei GOTO und GOSUB abzukürzen.

- Legen Sie DATA-Zeilen möglichst zusammen und ebenfalls in möglichst wenigen Zeilen an.

Leider vertragen sich (fast) alle diese Maßnahmen nicht mit der geforderten Übersicht über das Programm und sollten deshalb teilweise möglichst erst dann durchgeführt werden, wenn das Programm 'läuft'. Dieses 'speed up' Ihres Programms sollte also sein letzter Schliff sein.

Der zweite Punkt, das Ausführen von Maschinenprogrammen, ist natürlich etwas schwieriger, stellt aber bei vielen nicht arithmetischen (= nicht mathematischen) Prozessen eine sinnvolle und die effektivste Maßname dar, um Programme zu beschleunigen. Das Maschinenprogramm steht dazu in DATA-Zeilen und wird vor der Ausführung des eigentlichen Basicprogramms durch READ ausgelesen und in den jeweiligen Speicherbereich gePOKEt, in dem das Programm laufen soll, das später durch SYS aufgerufen wird (s. Zeichenformer, Spriteeditor und die vielen Beispiele, die von Assemblerprogrammen unterstützt werden).

Am eindrucksvollsten zeigt sich der Nutzen dieser Technik bei bestimmten graphischen Routinen (s. Graphik-Paket). Hier sollen die in Basic wohl zeitaufwendigsten Arbeiten mit der Graphik kurz durch entsprechende Assemblerroutinen ersetzt werden. Es sind dies: das Löschen der Graphik und die Farbgebung:

Graphik löschen

```
100 FOR I = 51200 TO 51221
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 169, 32,133,254,160,  0,132,253,162, 32,152,145
130 DATA 253,200,208,251,230,254,202,208,246, 96
140 IF S <> 3772 THEN PRINT "FEHLER IN DATAS !!" : END
150 PRINT "OK"
```

Farbe setzen

```
100 FOR I = 51222 TO 51261
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA  32,241,183,134,151,162,  3,169,  4,133,254,160
130 DATA   0,132,253,132,  2,165,151,145,253,200,196,  2
140 DATA 208,249,230,254,202,240,  3, 16,242, 96,162,232
150 DATA 134,  2,208,235
160 IF S <> 5970 THEN PRINT "FEHLER IN DATAS !!" : END
170 PRINT "OK"
```

Diese beiden Einleseroutinen können Sie in Ihre Programme einbauen, wenn Sie sich nicht die Mühe gemacht haben, das Graphik-Aid in Ihren Rechner zu tippen. Die Syntax der beiden neuen Graphikbefehle lautet:

```
SYS 51200 : REM GRAPHIK LOESCHEN
SYS 51222,16*PF+HF : REM FARBE SETZEN
```

Dabei bedeuten:

PF: Punktfarbe
HF: Hintergrundfarbe

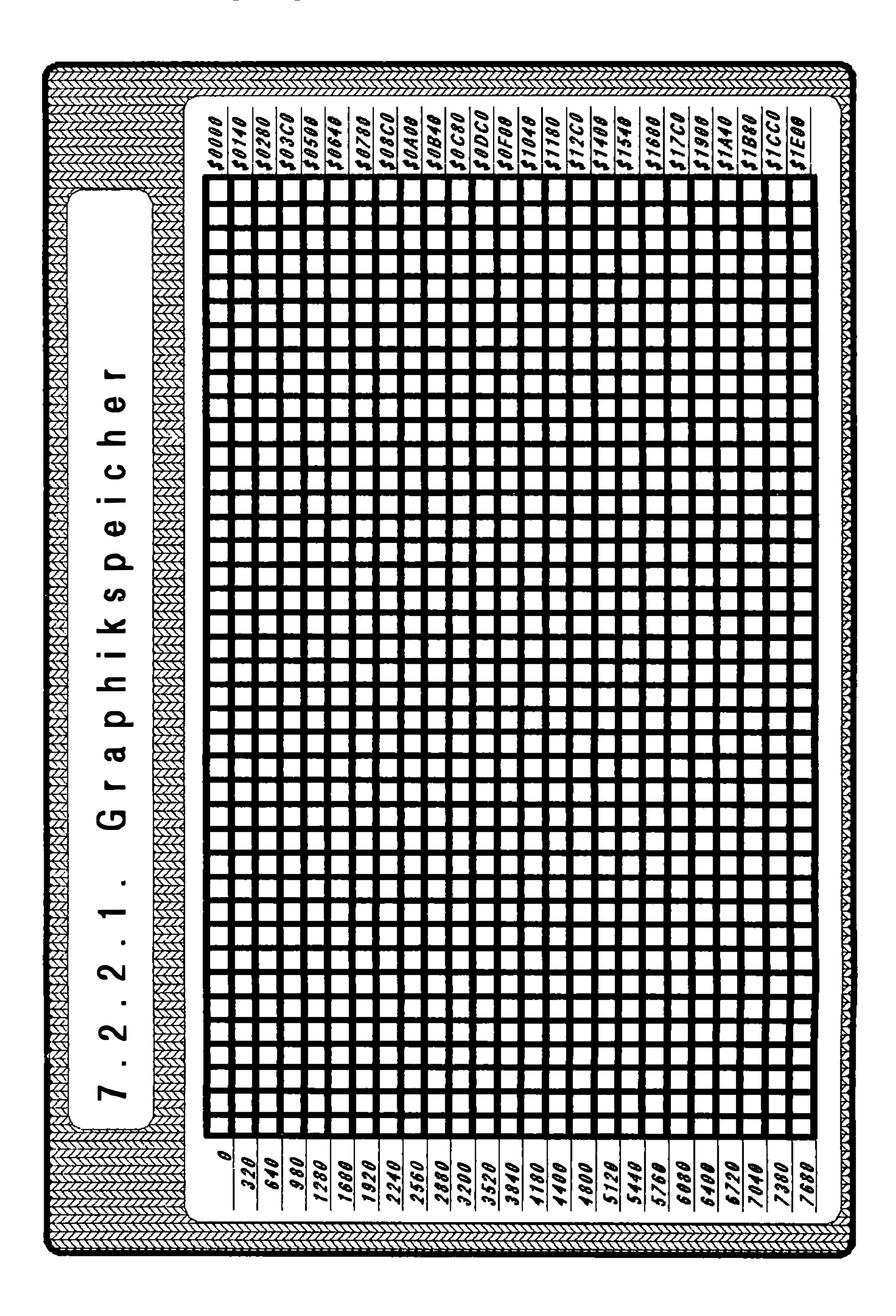
Wie sie anzuwenden sind, zeigen Ihnen die Erläuterungen in Kapitel 5.7.

7.2.2 Graphikspeicheraufbau

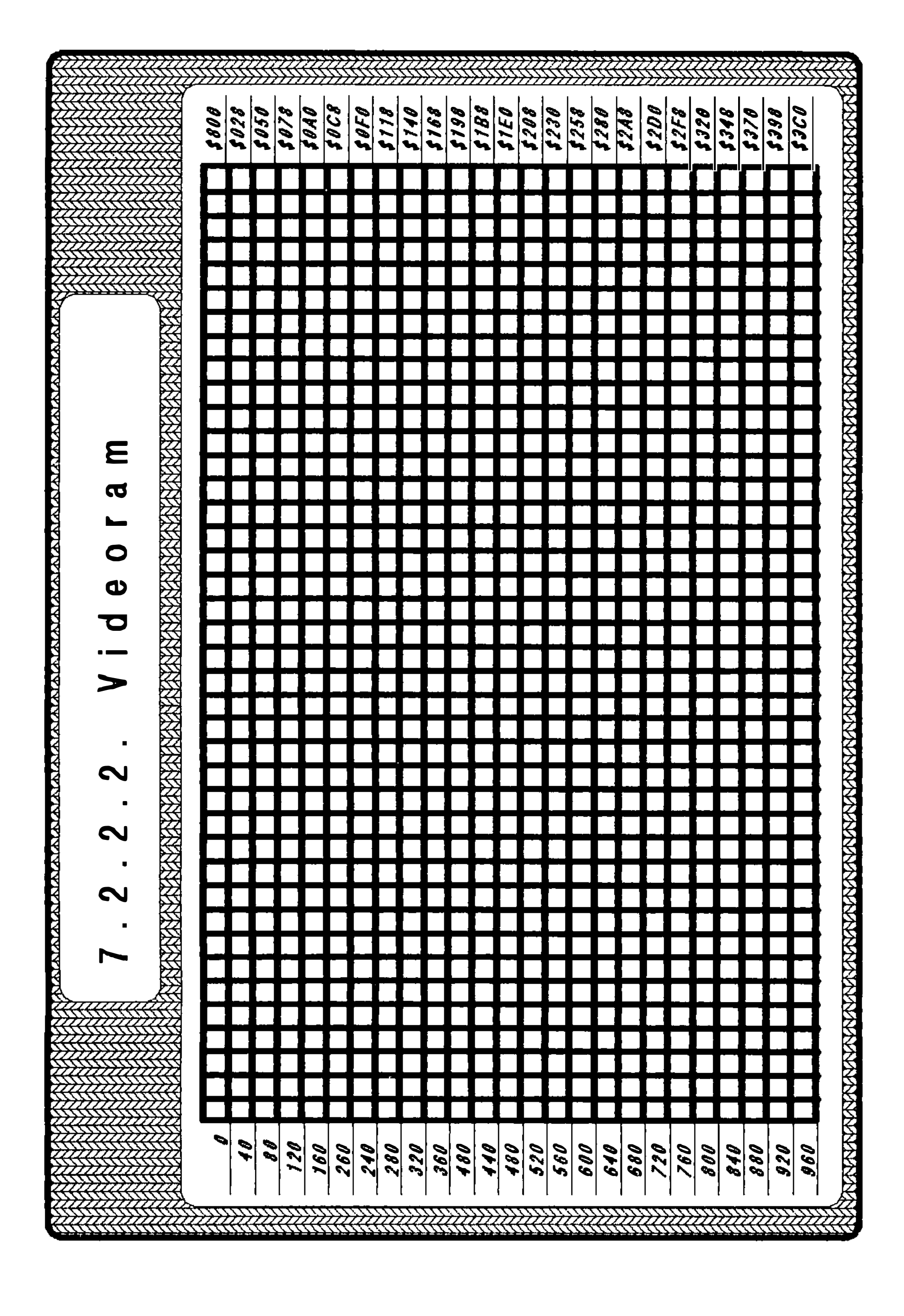
Im folgenden wird Ihnen der Aufbau der zwei wichtigsten Graphik-Speicherfunktionen dargestellt: Videoram und Graphikspeicher. Zu den am Rand angegebenen relativen Adressen der jeweiligen Zeilenanfänge müssen Sie stets noch die Basisadresse hinzuzählen. Diese hängt von der Lage des Speichers ab und lautet nach dem Einschalten für den Videoram: 1024 ($0400). Der Farbram besitzt den gleichen Aufbau wie der Videoram und hat die Basisadresse: 55296 ($D800).

Bei dem Schaubild für den Graphikspeicher müssen Sie jeden Kasten noch einmal in 8x8 Kästchen unterteilen, um die einzelnen Punkte zu symbolisieren. Den genauen Aufbau entnehmen Sie bitte dem 3. Kapitel.

7.2.2.1 Graphikspeicher



7.2.2.2 Video - RAM



7.2.3 Farbtabelle

Der Commodore 64 besitzt in allen verfügbaren Modi 16 Farben. Jeder dieser Farben ist ein sogenannter Farbcode zugeordnet, der in die entsprechenden Register gePOKEt wird, die für Farbangelegenheiten zuständig sind. Gleichzeitig können Sie aber auch alle 16 Farben im Textmodus für das Aussehen der Zeichen von Hand (Tastatur) aus anwählen. Dementsprechend gibt es für jede Farbe einen ASCII-Code, der die Bestimmung der Textfarbe auch von Programmen aus ermöglicht. In PRINT-Statements erscheinen die typischen Graphikzeichen. Die folgende Tabelle vereinigt alle Ansteuerungsmöglichkeiten übersichtlich zum schnellen Nachschlagen.

Taste	ASCII		Ausgabe	Codes		Farbe
	Dez	Hex		Dez	Hex	
<ctrl> 1	144	$90		00	$00	schwarz
<ctrl> 2	005	$05		01	$01	weiß
<ctrl> 3	028	$1C		02	$02	rot
<ctrl> 4	159	$9F		03	$03	zyanblau
<ctrl> 5	156	$9C		04	$04	violett
<ctrl> 6	030	$1E		05	$05	grün
<ctrl> 7	031	$1F		06	$06	blau
<ctrl> 8	158	$9E		07	$07	gelb
<C=> 1	129	$81		08	$08	orange
<C=> 2	149	$95		09	$09	braun
<C=> 3	150	$96		10	$0A	hellrot
<C=> 4	151	$97		11	$0B	dunkelgrau
<C=> 5	152	$98		12	$0C	mittelgrau
<C=> 6	153	$99		13	$0D	hellgrün
<C=> 7	154	$9A		14	$0E	hellblau
<C=> 8	155	$9B		15	$0F	hellgrau

7.2.4 Bildschirmcodes

Jedem Zeichen des Bildschirms ist ein bestimmter ASCII-Code, aber auch ein Bildschirmcode zugeordnet. Letzterer ist der Code, mit dem das Zeichen im Videoram abgespeichert wird. Wollen Sie also ein Zeichen direkt in diesen Textspeicher POKEn, so verwenden Sie jene Werte. Die Codes für die inversen Zeichen erhalten Sie, indem Sie 128 zu denen der normalen Zeichen hinzuaddieren.

Codes	ASCII	Zeichen	
		Satz1	Satz2
0	64	@	@
1	65	A	a
2	66	B	b
3	67	C	c
4	68	D	d
5	69	E	e
6	70	F	f
7	71	G	g
8	72	H	h
9	73	I	i
10	74	J	j
11	75	K	k
12	76	L	l
13	77	M	m
14	78	N	n
15	79	O	o
16	80	P	p
17	81	Q	q
18	82	R	r
19	83	S	s
20	84	T	t
21	85	U	u
22	86	V	v
23	87	W	w
24	88	X	x
25	89	Y	y
26	90	Z	z
27	91	[	[
28	92	£	£
29	93	]	]
30	94	↑	↑
31	95	←	←
32	32		
33	33	!	!
34	34	"	"
35	35	#	#
36	36	$	$
37	37	%	%
38	38	&	&
39	39	'	'

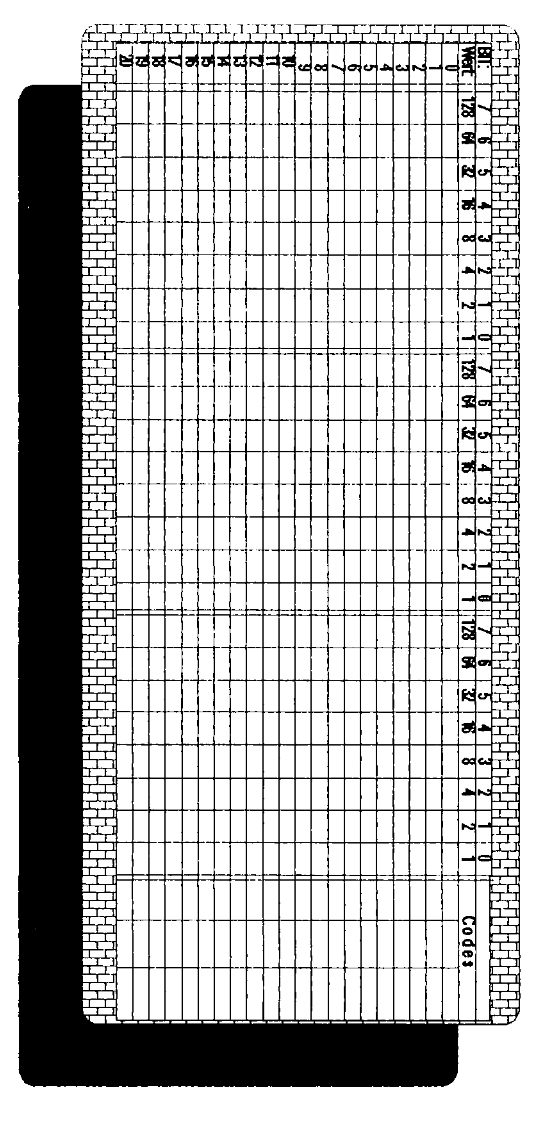
Codes	ASCII	Zeichen	
		Satz1	Satz2
40	40	(	(
41	41	)	)
42	42	*	*
43	43	+	+
44	44	,	,
45	45	-	-
46	46	.	.
47	47	/	/
48	48	0	0
49	49	1	1
50	50	2	2
51	51	3	3
52	52	4	4
53	53	5	5
54	54	6	6
55	55	7	7
56	56	8	8
57	57	9	9
58	58	:	:
59	59	;	;
60	60	<	<
61	61	=	=
62	62	>	>
63	63	?	?
64	96	─	─
65	97	♠	A
66	98	│	B
67	99	─	C
68	100	─	D
69	101	‾	E
70	102	─	F
71	103	│	G
72	104	│	H
73	105	╮	I
74	106	╰	J
75	107	╯	K
76	108	L	L
77	109	╲	M
78	110	╱	N
79	111	┌	O

7.2.5 Dez/Hex/Dual-Konversionstabelle

		%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%
		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
%0000 0000	0	$00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
%0001 0000	16	$10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
%0010 0000	32	$20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
%0011 0000	48	$30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
%0100 0000	64	$40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
%0101 0000	80	$50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
%0110 0000	96	$60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
%0111 0000	112	$70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
%1000 0000	128	$80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
%1001 0000	144	$90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
%1010 0000	160	$A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
%1011 0000	176	$B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
%1100 0000	192	$C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
%1101 0000	208	$D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
%1110 0000	224	$E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
%1111 0000	240	$F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

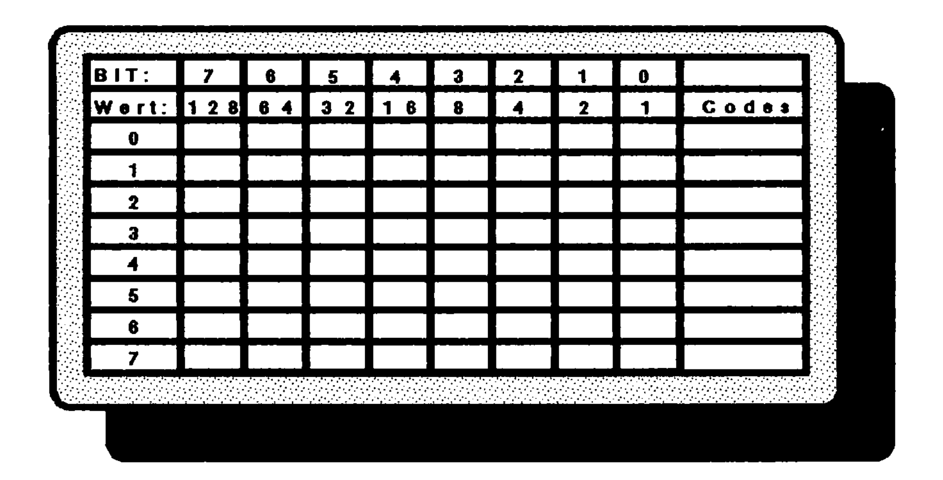
7.2.6 Spriteentwurfsblatt

Mit Hilfe dieses Formblattes ist es Ihnen möglich auf einfache Art und Weise Sprites zu erstellen, indem Sie die einzelnen Kästchen, die jeweils einen Punkt darstellen, ausmalen und die oben stehenden Werte zu einem Byte (drei Bytes pro Zeile) zusammenzählen. Multicolorsprites erstellen Sie, indem Sie für einen Punkt zwei nebeneinander liegende Kästchen (möglichst die 3 verschiedenen Multicolorfarben auch in drei verschiedenen Farben zeichnen) ausmalen.



7.2.7 Zeichenentwurfsblatt

Ebenso wie beim Spriteentwurfsblatt können Sie auch mit der folgenden Tabelle Multicolorzeichen erstellt werden, indem Sie für einen Punkt jeweils zwei Kästchen ausmahlen.



BIT:	7	6	5	4	3	2	1	0	
Wert:	128	64	32	16	8	4	2	1	Codes
0									
1									
2									
3									
4									
5									
6									
7									

7.2.8 VIC-Register-Übersicht

Register		Adresse		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Dez	Hex	Dez	Hex	128	64	32	16	8	4	2	1
00	$00	53248	$D000	Sprite 0 --- x - Koordinate (Bits 0-7)(0-255)							
01	$01	53249	$D001	Sprite 0 --- y - Koordinate (0-255)							
02	$02	53250	$D002	Sprite 1 --- x - Koordinate (Bits 0-7)(0-255)							
03	$03	53251	$D003	Sprite 1 --- y - Koordinate (0-255)							
04	$04	53252	$D004	Sprite 2 --- x - Koordinate (Bits 0-7)(0-255)							
05	$05	53253	$D005	Sprite 2 --- y - Koordinate (0-255)							
06	$06	53254	$D006	Sprite 3 --- x - Koordinate (Bits 0-7)(0-255)							
07	$07	53255	$D007	Sprite 3 --- y - Koordinate (0-255)							
08	$08	53256	$D008	Sprite 4 --- x - Koordinate (Bits 0-7)(0-255)							
09	$09	53257	$D009	Sprite 4 --- y - Koordinate (0-255)							
10	$0A	53258	$D00A	Sprite 5 --- x - Koordinate (Bits 0-7)(0-255)							
11	$0B	53259	$D00B	Sprite 5 --- y - Koordinate (0-255)							
12	$0C	53260	$D00C	Sprite 6 --- x - Koordinate (Bits 0-7)(0-255)							
13	$0D	53261	$D00D	Sprite 6 --- y - Koordinate (0-255)							
14	$0E	53262	$D00E	Sprite 7 --- x - Koordinate (Bits 0-7)(0-255)							
15	$0F	53263	$D00F	Sprite 7 --- Y - Koordinate (0-255)							

Register		Adresse		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Dez	Hex	Dez	Hex	128	64	32	16	8	4	2	1
16	$10	53264	$D010	Sprite 0-7 --- x - Koordinaten (Bit 8)(*256)							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
17	$11	53265	$D011	Raster	extend.	Hi-Res-	Bild	25/24	y-Scrolling		
				Bit 8	Color	Graphik	aus	Zeilen			
18	$12	53266	$D012	Rasterzeile (aktuelle/Vorgabe)(Bits 0-7)(0-255)							
19	$13	53267	$D013	Lightpen --- x - Koordinate (0-255)							
20	$14	53268	$D014	Lightpen --- y - Koordinate (0-255)							
21	$15	53269	$D015		Sprite --- an / aus						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
22	$16	53270	$D016	nicht benutzt			Multi-	40/38	x-Scrolling		
							color	Spalten			
23	$17	53271	$D017	Sprite --- y - Vergrößerung							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
24	$18	53272	$D018	Bildschirmspeicheradr. (*1024)				Zeichengen.adr.(*2048)			nicht
				Bit 13	Bit 12	Bit 11	Bit 10	Bit 13	Bit 12	Bit 11	benutzt
25	$19	53273	$D019	IRQ-	nicht benutzt			Light-	Sp.-Sp	Sp.-Hg.	Raster
				Flag				pen	Kollis.	Kollis.	zeilen
26	$1A	53274	$D01A	nicht benutzt				Light-	Sp.-Sp	Sp.-Hg.	Raster
								pen	Kollis.	Kollis.	zeilen
27	$1B	53275	$D01B		Sprite - Hintergrund - Priorität						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
28	$1C	53276	$D01C		Multicolor - Sprites						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
29	$1D	53277	$D01D		Sprite --- x - Vergrößerung						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0

Register		Adresse		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Dez	Hex	Dez	Hex	128	64	32	16	8	4	2	1
30	$1E	53278	$D01E	Sprite - Sprite - Kollision							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
31	$1F	53279	$D01F	Sprite - Hintergrund - Kollision							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
32	$20	53280	$D020	Rahmenfarbe (0-15)							
33	$21	53281	$D021	Hintergrundfarbe Nr. 0 (0-15)							
34	$22	53282	$D022	Hintergrundfarbe Nr. 1 (0-15)							
35	$23	53283	$D023	Hintergrundfarbe Nr. 2 (0-15)							
36	$24	53284	$D024	Hintergrundfarbe Nr. 3 (0-15)							
37	$25	53285	$D025	Gemeinsame Sprite-Farbe Nr. 0 in Multicolor (0-15)							
38	$26	53286	$D026	Gemeinsame Sprite-Farbe Nr. 1 in Multicolor (0-15)							
39	$27	53287	$D027	Farbe für Sprite Nr. 0 (0-15)							
40	$28	53288	$D028	Farbe für Sprite Nr. 1 (0-15)							
41	$29	53289	$D029	Farbe für Sprite Nr. 2 (0-15)							
42	$2A	53290	$D02A	Farbe für Sprite Nr. 3 (0-15)							
43	$2B	53291	$D02B	Farbe für Sprite Nr. 4 (0-15)							
44	$2C	53292	$D02C	Farbe für Sprite Nr. 5 (0-15)							
45	$2D	53293	$D02D	Farbe für Sprite Nr. 6 (0-15)							
46	$2E	53294	$D02E	Farbe für Sprite Nr. 7 (0-15)							

7.2.9 Weiterführende Literatur

Commodore 64:

Sally Greenwood Larsen
Sprite Graphics for the Commodore 64
Micro Text Publications
ISBN: 0-13-838144-5

Compute!'s First Book of Commodore 64
Compute! Books Publication
ISBN: 0-942386-20-5

C. Lorenz
Beherrschen Sie Ihren Commodore 64
Hofacker Verlag
ISBN: 3-88963-147-9

Stan Krute
Commodore 64 Graphics & Sound Programming
Tab Books Inc.
ISBN: 0-8306-0140-6

Shaffer
Commodore 64 Color Graphics: A Beginners Guide
The Book Company
ISBN:0-912003-06-5

Angerh./Brück./Eng./Gerits
64 intern
DATA BECKER GmbH
ISBN: 3-89011-000-2zln1
Graphik-Fachbücher:

Faux/Pratt
Computational Geometry for Design and Manufacture
Ellis Horwood Limited
ISBN: 0-85312-114-1

Hearn/Baker
Computer Graphics for the IBM Personal Computer
Prentice-Hall, Inc.
ISBN: 0-13-164335-5

Encarnacao
Computer Aided Design-Modelling, Systems Engineering, CAD-Systems
Springer-Verlag
ISBN: 0-387-10242-6

Encarnacao
Computer-Graphics - Programmierung und Anwendung von graphischen Systemen
R.Oldenbourg Verlag
ISBN: 3-486-34651-2

Brodlie
Mathematical Methods in Computer Graphics and Design
Academic Press
ISBN: 0-12-134880-6

Barnhill/Boehm
Surfaces in Computer Aided Geometric Design
North-Holland
ISBN: 0-444-86550-0

Myers
Microcomputer Graphics
Addison-Wesley Publishing Company
ISBN: 0-201-05092-7

Besonders zu empfehlen:

David F. Rogers
Procedural Elements for
Computer Graphiks
Mc Graw-Hill Book Company
ISBN 0-07-053534-5

Steven Harrington
Computer Graphics
A Programming Approach
Mc Graw-Hill Book Company
ISBN 0-07-026751-0

Günter Spur/Frank-Lothar Krause
CAD-Technik
Carl Hanser Verlag München Wien
ISBN 3-446-13897-8

Roy A. Plastock/Gordon Kalley
Theory and problems of
Computer Graphics
Mc Graw-Hill Book Company
ISBN 0-07-050326-5

Braun
Atari ST
3D Grafik Programmierung
DATA BECKER GmbH
ISBN 3-89011-130-0

Norbert Treitz
Spiele mit Computer Grafik
für Phantasie und Logik
Hagemann Verlag
ISBN 3-544-53003-1

7.3 Anhang zur Supergraphik

7.3.1 Kurzbeschreibung der Supergraphik-Befehle

A. Grundbefehle:

a) Zeichenmodi:

zm	Wirkung	Beispiel
0	Figur zeichnen	PLOT 0,160,100
1	Figur loeschen	PLOT 1,160,100
2	Figur invertieren	PLOT 2,160,100
3	Figur punktieren	PLOT 3,160,100
4	Graphikcursor bewegen	PLOT 4,160,100

b) Secundärbefehle:

C - Zeichnen mit Farbsetzung

Beispiel: PLOT C 1,100,150

B m, - Pinselwahl

Beispiel: PLOT B 45,,20,40

T var, -

```
PLOT                (zm),x,y
PLOT C              (zm),x,y
PLOT   B m,         (zm),x,y
PLOT C B m,         (zm),x,y
PLOT         T var, (zm),x,y
PLOT C       T var, (zm),x,y
PLOT   B m,  T var, (zm),x,y
PLOT C B m,  T var, (zm),x,y
```

c) Befehle:

PLOT

Befehl:	PLOT zm,x,y
Beispiel:	PLOT 1,100,199
Parameter:	zm: Zeichenmodus
	x: x-Koordinate
	y: y-Koordinate
Funktion:	Setzen (Loeschen etc.) eines Punktes

PLOT ... TO ...

Befehl:	PLOT zm,x1,y1 TO x2,y2 (TO x3,y3 TO ...)
Beispiel:	PLOT 0,100,199 TO 0,20
Parameter:	zm: Zeichenmodus
	x1: x-Startkoordinate
	y1: y-Startkoordinate
	x2: x-Zielkoordinate
	y2: y-Zielkoordinate
Funktion:	Setzen (Loeschen etc.) einer oder mehrerer Linien

PLOT TO ...

Befehl:	PLOT zm, TO x2,y2 (TO x3,y3 TO ...)
Beispiel:	PLOT 2, TO 100,199
Parameter:	zm: Zeichenmodus
	x2: x-Zielkoordinate
	y2: y-Zielkoordinate
Funktion:	Setzen (Loeschen etc.) einer Linie vom Graphikcursor

CIRCLE

Befehl:	CIRCLE zm,,xm,ym,xr,yr
Beispiel:	CIRCLE 0,,160,100,80,90
Parameter:	zm: Zeichenmodus
	xm: x-Mittelpunktskoordinate
	ym: y-Mittelpunktskoordinate
	xr: x-Radius
	yr: y-Radius
Funktion:	Setzen (Loeschen etc.) einer Ellipse, Kreis

Bögen:

Befehl:	CIRCLE zm,(sl),xm,ym,xr,yr,wa,we
Beispiel:	CIRCLE 0,2,160,100,80,90,30,130
Parameter:	zm: Zeichenmodus
	sl: Seitenlänge (0-255)
	xm: -Mittelpunktskoordinate
	ym: -Mittelpunktskoordinate
	xr: -Radius
	yr: y-Radius
	wa: Anfangswinkel (0-360)
	we: Endwinkel (0-360)
Funktion:	Setzen (Loeschen etc.) von Kreis-, Ellipsen- oder Vieleck-Bögen

---

DRAW

Befehl:	DRAW zm,str$ ON x,y
Beispiel:	DRAW 2,A$ ON 160,100
Parameter:	zm: Zeichenmodus
	str$: Definitionsstring
	x: x-Koordinate
	y: y-Koordinate
Funktion:	Setzen (Loeschen etc.) einer frei definierbaren Figur

0 = hoch
1 = rechts
2 = runter
3 = links

Befehl:	DRAW zm,str$
Beispiel:	DRAW 2,A$
Parameter:	zm: Zeichenmodus
	str$: Definitionsstring
Funktion:	Setzen (Loeschen etc.) einer frei definierbaren Figur Graphikcursor aus

---

FRAME

Befehl: FRAME zm,d,x1,y1 TO x2,y2
Beispiel: FRAME 0,10,100,100 TO 200,150
Parameter: zm: Zeichenmodus
d: Rahmendicke
x1: x-Eckkoordinate 1
y1: y-Eckkoordinate 1
x2: x-Eckkoordinate 2
y2: y-Eckkoordinate 2
Funktion: Setzen (Loeschen etc.) eines Rahmens definierter Dicke

---

FILL

Befehl: FILL zm,x1,y1 TO x2,y2
Beispiel: FILL 0,100,100 TO 200,150
Parameter: zm: Zeichenmodus
x1: x-Eckkoordinate 1
y1: y-Eckkoordinate 1
x2: x-Eckkoordinate 2
y2: y-Eckkoordinate 2
Funktion: Setzen (Loeschen etc.) eines Feldes

TEXT

Befehl: TEXT zm,str$,x,y,m
Beispiel: TEXT 0,"HALLO",100,130,0
Parameter: zm: Zeichenmodus
str$: zu schreibender Text
x: x-Koordinate
y: y-Koordinate
m: Textmodus (0,1)
Funktion: Setzen (Loeschen etc.) eines Textes als Graphik

m = 0: Groß/Graphikmodus
m = 1: Groß/Kleinschrift

PAINT

Befehl:	PAINT zm,x,y	
Beispiel:	PAINT ,160,100	
Parameter:	zm:	Zeichenmodus
	x:	x-Koordinate des Testpunktes
	y:	y-Koordinate des Testpunktes
Funktion:	Ausfüllen einer beliebigen geschlossenen Fläche	

B. Weitere Graphikbefehle:

GMODE

Befehl:	GMODE (s)(,m)(,or,ur)
Beispiel:	GMODE 0,1
oder:	GMODE 2,7,30,70
Parameter:	s: Seitenmodus (s.u.)
	m: Graphikmodus (s.u.)
	or: oberer Textrand bei m=7/8
	ur: unterer Textrand bei m=7/8
Funktion:	bestimmen des Graphikmodus

s=0:	Graphikseite 1 wird angezeigt und befehligt
s=1:	Graphikseite 1 wird angezeigt, Seite 2 aber befehligt
s=2:	Graphikseite 2 wird angezeigt und befehligt
s=3:	Graphikseite 2 wird angezeigt, Seite 1 aber befehligt

m=0:	Textmodus bzw. LGR werden angezeigt und auch befehligt
m=1:	HGR wird angezeigt und befehligt
m=2:	MC wird angezeigt und befehligt
m=3:	HGR wird angezeigt, LGR jedoch gefehligt, d.h. die Graphikbefehle beziehen sich auf die unsichtbare Low-Graphik
m=4:	MC wird angezeigt, jedoch LGR befehligt
m=5:	LGR und normaler Text werden angezeigt, die Befehle gelten aber für die hochauflösende Graphik
m=6:	LGR bzw. Text werden angezeigt, MC jedoch befehligt
m=7:	Gemischte Text- und Graphikanzeige, LGR wird befehligt. Im Anschluß wird die Angabe der Fensterposition verlangt
m=8:	Gemischte Text- und Graphikanzeige, HGR wird befehligt; Angabe der Fensterposition

GCLEAR

Befehl:	GCLEAR (m)(,x1,y1 TO x2,y2)
Beispiel:	GCLEAR
	GCLEAR 255
	GCLEAR 4,30,40 TO 50,70
Parameter:	m: Löschmaske
	x1: x-Koordinate Fenster oben links
	y1: y-Koordinate Fenster oben links
	x2: x-Koordinate Fenster unten rechts
	y2: y-Koordinate Fenster unten rechts
Funktion:	Löschen des Graphikbildschirms bzw. eines Bildschirmfensters

INVERS

Befehl:	INVERS (m)(,x1,y1 TO x2,y2)
Beispiel:	INVERS
	INVERS 240
	INVERS 240,20,30 TO 60,70
Parameter:	m: Invertierungsmaske
	x1: x-Koordinate Fenster oben links
	y1: y-Koordinate Fenster oben links
	x2: x-Koordinate Fenster unten rechts
	y2: y-Koordinate Fenster unten rechts
Funktion:	Invertieren des Graphikbildes oder eines Graphikfensters mit 256 verschieden Masken

GCOMB

Befehl:	GCOMB m (,x1,y1 TO x2,y2 (TO x3,y3))
Beispiel:	GCOMB 1
	GCOMB 2,20,30 TO 50,120
	GCOMB 3,40,10 TO 90,130 TO 30,50
Parameter:	m: Verknüpfungsmodus (0-7)
	x1: x-Koordinate Fenster oben links
	y1: y-Koordinate Fenster oben links
	x2: x-Koordinate Fenster unten rechts
	y2: y-Koordinate Fenster unten rechts
	x3: x-Koordinate Zielfenster oben links
	y3: y-Koordinate Zielfenster oben links
Funktion:	Verknüpfung beider Graphikseiten, zweier Bildschirmfenster, Verschiebung eines Graphikfensters

m=0, m=4: UND-Verknüpfung zweier Seiten
m=1, m=5: ODER-Verknüpfung zweier Seiten
m=2, m=6: EXKLUSIV-ODER-Verknüpfung der Seiten
m=3, m=7: Copieren eines Bildschirmfensters in die andere Seite

TRANS

Befehl:	TRANS m
Beispiel:	TRANS 1
Parameter:	m: Zeichensatz (0/1)
Funktion:	Copieren der Textseite in die Graphik

m=0: Groß-/Graphiksatz
m=1: Groß-/Kleinschrift

GMOVE

Befehl:	GMOVE m,za,ze
Beispiel:	GMOVE 0,10,20
Parameter:	m: Verschiebemodus (0-3)
	za: Anfangszeile (0-24)
	ze: Endzeile (0-24)
Funktion:	Horizontales Rollen/Verschieben des Bildschirms

m=0: Links rollen
m=1: Rechts rollen
m=2: Links verschieben
m=3: Rechts verschieben

SCALE=

Befehl:	SCALE= r,s
Beispiel:	SCALE= 1,5
Parameter:	r: Rotationswinkel
	s: Größe
Funktion:	Rotieren und Vergrößern einer DRAW-Figur

B. Farbbefehle:

COLOR=

Befehl:	COLOR= r,h
Beispiel:	COLOR= 2,6
Parameter:	r: Rahmenfarbe (0-15)
	h: Hintergrundfarbe (0-15)
Funktion:	Festlegen der Rahmen- und Hintergrundfarbe

SCOL=

Befehl:	SCOL= n,f
Beispiel:	SCOL= 1,14
Parameter:	n: Farbregister (0-3)
	f: Farbe
Funktion:	Festlegen der Zeichenfarbe für den gesamten Bildschirm.

PCOL=

Befehl:	PCOL= n,f
Beispiel:	PCOL= 1,14
Parameter:	n: Farbregisternummer/Registermodus (0-4)
	f: Farbnummer (0-15)/Farbregisternummer (0-3)
Funktion:	Festlegen der Zeichenfarbe

D. Eingabe/Ausgabebefehle:

GSAVE

Befehl: GSAVE (LGCF),s,"filename"(,ga)
Beispiel: GSAVE GCF,1,"MULTICOLOR",8
Parameter: LGCF : Formatsteuerung (s.u.)
s: Angesteuerte Graphikseite
ga: Geräteadresse
Funktion: Speichern der Graphik, Farbe oder Text

L: Low-Graphik (Text) (1000 Bytes)
G: Graphikspeicher (8000 Bytes)
C: Farbvideoram (1000 Bytes)
F: Farbram (1000 Bytes)

GLOAD

Befehl: GLOAD (LGCF),s,"filename"(,ga)
Beispiel: GLOAD GCF,1,"MULTICOLOR",8
Parameter: LGCF: Formatsteuerung (s.u.)
s: Angesteuerte Graphikseite
ga: Geräteadresse
Funktion: Laden der Graphik, Farbe oder Text

HCOPY

Befehl: HCOPY# n
Beispiel: HCOPY# 1
Parameter: n: logische Filenummer
Funktion: Fertigen einer Hardcopy der Graphikanzeige

- HCOPYD cf,df - Farbzuordnung (Nur Farbdrucker)
- HCOPYC#n,d - HCOPY nur für Farbdrucker

E. Sprite-Befehle:

SREAD

Befehl: SREAD str$
Beispiel: SREAD A$
Parameter: str$: Zielstringvariable
Funktion: Einlesen von 63 Spritedaten

SDEFINE

Befehl: SDEFINE s,str$
Beispiel: SDEFINE 9,A$
Parameter: s: Spritenummer (0-15)
str$: Definitionsstring
Funktion: Spritedefinition-Einschalten/Ausschalten

SPOWER

Befehl: SPOWER or,ur
Beispiel: SPOWER 100,150
Parameter: or: oberer Fensterrand
ur: unter Fensterrand
Funktion: Zuschalten des zweiten Spritesatzes

Befehl: SPOWER
Beispiel: SPOWER
Parameter: ---
Funktion: Abschalten des zweiten Spritesatzes

SMODE

Befehl: SMODE s,m,f1 (,f2,f3)
Beispiel: SMODE 1,3,7
oder SMODE 1,8,7,2,14
Parameter: s: Spritenummer
m: Spritemodus (s.u.)
f1: Farbe 1 (individuell)(0-15)
f2: Farbe 2 (Multicolor-einheitlich)(0-15)
f3: FArbe 3 (Multicolor-einheitlich)(0-15)
Funktion: Bestimmen der Spriteeigenschaften

m MC	m HGR	Vergrößerung x-Richtung	y-Richtung	Priorität vor dem Hintegrund
8	0	nein	nein	ja
9	1	ja	nein	ja
10	2	nein	ja	ja
11	3	ja	ja	ja
12	4	nein	nein	nein
13	5	ja	nein	nein
14	6	nein	ja	nein
151	7	ja	ja	nein

SSET

Befehl: SSET s,x,y (,sp)
Beispiel: SSET 1,100,120
oder SSET 1,100,120,4
Parameter:
s: Spritenummer
x: Sprite-x-Koordinate
y: Sprite-y-Koordinate
sp: Geschwindigkeit für Spritebewegung
Funktion: Positionieren eines Sprites

SSET...TO...

Befehl: SSET s,x1,y1 TO x2,y2 (,sp)
Beispiel: SSET 1,100,120 TO 50,40
oder SSET 1,100,120 TO 50,40,4
Parameter:
s: Spritenummer
x1: Sprite-x-Startkoordinate
y1: Sprite-y-Startkoordinate
x2: Sprite-x-Zielkoordinate
y2: Sprite-y-Zielkoordinate
sp: Geschwindigkeit für Spritebewegung
Funktion: Bewegen eines Sprites von x1,y1 nach x2,y2

SSET TO...

Befehl:	SSET s TO x2,y2 (,sp)
Beispiel:	SSET 1 TO 50,40
oder	SSET 1 TO 50,40,4
Parameter:	s: Spritenummer
	x2: Sprite-x-Zielkoordinate
	y2: Sprite-y-Zielkoordinate
	sp: Geschwindigkeit für Spritebewegung
Funktion:	Bewegen eines Sprites von der momentanen Position nach x2,y2

SWAIT

Befehl:	SWAIT s
Beispiel:	SWAIT 2
Parameter:	s: Spritenummer
Funktion:	Warten bis Sprite s steht

IF# ... THEN ...

Befehl:	IF# par THEN ...
Beispiel:	IF# U,1 THEN ...
Parameter:	par: Secundärfunktion
Funktion:	Bedingtes Verzweigen unter definierten Bedingungen

IF# U p THEN ...:
Verzweigung bei Joystick in Port p nach oben gedrückt wird.

IF# D p THEN ...:
Verzweigung bei Joystick nach unten.

IF# R p THEN ...:
Verzweigung bei Joystick nach rechts.

IF# L p THEN ...:
Verzweigung bei Joystick nach links.

IF# B p THEN ...:
Verzweigung, wenn der Feuerknopf (Button) Ihres Joysticks gedrückt ist.

IF# CN THEN ...:
Verzweigung, wenn gerade im Augenblick eine Kollision eines Sprites mit einem anderen Sprite oder mit einem Hintergrundzeichen stattfindet.

IF# C THEN ...:
Einschalten des Kollisionsinterrupt-Modus.

IF# CC THEN ...:
Ausschalten des Kollisionsinterrupt-Mouds.

IRETURN

Befehl:	IRETURN
Beispiel:	IRETURN
Parameter:	///
Funktion:	Beenden der Unterbrechungsroutine

F. Ton-Befehle:

VOLUME=

Befehl: VOLUME= v
Beispiel: VOLUME= 15
Parameter: v: Lautstärke
Funktion: Einstellen der Lautstärke

SOUND

Befehl: SOUND st,w,a,d,s,r (,f (,p))
Beispiel: SOUND 1,3,12,11,5,5
oder SOUND 1,3,12,11,5,5,1
oder SOUND 1,3,12,11,5,5,1,100
Parameter: st: Stimme (1-3)
w: Wellenform (0-8)
a: Attack (Anschwellzeit)
d: Decay (Abschwellzeit)
s: Sustain (Haltelautstärke)
r: Release (Ausklingzeit)
f: Filter an/aus (0-1)
p: Pulsrate (Tastverhältnis)(0-4095)
Funktion: Einstellen des Klangbildes

w=0: Stimme ist stumm
w=1: Dreiecksschwingung
w=2: Sägezahn
w=3: Dreieck und Sägezahn
w=4: Rechteck
w=5: Dreieck und Rechteck
w=6: Sägezahn und Rechteck
w=7: Dreieck, Sägezahn und Rechteck
w=8: Rauschen

FILTER

Befehl: FILTER fa (,ff,rf)
Beispiel: FILTER 4
oder FILTER 4,1000,14
Parameter: fa: Filterart (0-7)
ff: Filtergrenzfrequenz (0-2047)
rf: Filterresonanzfrequenz (0-15)
Funktion: Einstellen der Filter

fa=0: kein Filter an
fa=1: Tiefpass
fa=2: Bandpass
fa=3: Tief- und Bandpass
fa=4: Hochpass
fa=5: Hoch- und Tiefpass (Bandsperre)
fa=6: Hoch- und Bandpass
fa=7: Hoch-, Band- und Tiefpass (Bandsperre)

TUNE

Befehl:	TUNE st,l,n,o (,v)	
Beispiel:	TUNE 2,30,3,4	
oder	TUNE 2,30,3,4,14	
Parameter:	st:	Stimme (1-3)
	l:	Länge des Tones (0-65535)
	n:	Note (0-12)
	o:	Oktave (0-7)
	v:	Verstimmung (0-255)
Funktion:	Spielen eines Tones	

```
c ,cis, d ,dis, e , f ,fis, g ,gis, a ,ais, h
1   2   3   4   5   6   7   8   9  10  11  12
```

v=1-127: Verstimmung um v nach oben
v=255-128: Verstimmung um 256-v nach unten

G. Utilities:

DIRECTORY

Befehl:	DIRECTORY
Beispiel:	DIRECTORY
Parameter:	---
Funktion :	Anzeigen des Disketteninhaltsverzeichnisses ohne Programmverlust

MERGE

Befehl:	MERGE "filename",ga
Beispiel:	MERGE "HUSTENBONBON",8
Parameter:	ga: Geräteadresse
Funktion:	Anhängen eines Basicprogrammes von Diskette an ein im Speicher befindliches

RENUM

Befehl:	RENUM sz,a
Beispiel:	RENUM 10,50
Parameter:	sz: Startzeile der neuen Version
	a: Abstand der Basiczeilen (1-255)
Funktion:	Umnumerierung eines Basicprogrammes

DTASET

Befehl:	DTASET zn
Beispiel:	DTASET 100
Parameter:	zn: Zeilennummer
Funktion:	Setzen des DATA-Zeigers auf eine bestimmte Zeile

KEY

Befehl:	KEY ft,"definition"
Beispiel:	KEY 2,"LIST"+CHR$(13)
Parameter:	ft: angewählte Funktionstaste (1-8)
	"definition": 16 Zeichen-Wort
Funktion:	Umprogrammieren der Funktionstasten

PADDLE

Befehl:	PADDLE var,pn
Beispiel:	PADDLE A,2
Parameter:	var: Zielvariable
	pn: Paddlenummer (1-4)
Funktion:	Auslesen der A/D-Wandler (Paddle-Werte)

POS=

Befehl:	POS= s,z
Beispiel:	POS= 10,15
Parameter:	s: Spalte (0-39)
	z: Zeile (0-24)
Funktion:	Positionieren des Text-Cursors

7.3.2 Befehlsabkürzungen:

Zusätzlich zu den bekannten Kürzeln (s. Handbuch) können Sie für die Befehle Ihrer Supergraphik 64 noch folgende Abkürzungen verwenden (zusätzliche Buchstaben werden an das Kürzel angehängt). ^ bedeutet, dieser Buchstabe wird gleichzeitig mit der <shift>-Taste gedrückt. Weiterhin haben Sie hier eine Übersicht über die Befehlstokens, mit denen die Supergraphik jeden Befehl im Programmspeicher abgekürzt vermerkt (s. auch Supergraphik-Listing):

Befehl	Kürzel	Tokens
CIRCLE	C^I	$F2
COLOR=	C^O	$F5
DIRECTORY	D^I	$D7
DRAW	D^R	$ED
DTASET	D^T	$DA
FILTER	F^I	$E3
FILL	FI^L	$EE
FRAME	F^R	$EF
GCLEAR	GC^L	$EA
GCOMB	G^C	$D9
GLOAD	G^L	$F9
GMODE	G^M	$E9
GMOVE	GMO^V	$EB
GSAVE	G^S	$F8
HCOPY	H^C	$FA
IF#	I^F	$FC
INVERS	I^N	$F0
IRETURN	I^R	$FB
KEY	K^E	$DD
MERGE	M^E	$DB
PADDLE	P^A	$F3
PAINT	PA^I	$FD
PCOL=	P^C	$F7
PLOT	P^L	$EC
RENUM	R^E	$DC
SCALE=	S^C	$F4
SCOL=	SC^O	$F6
SDEFINE	S^D	$E5
SMODE	S^M	$E8
SOUND	S^O	$E1
SPOWER	S^P	$D8
SREAD	S^R	$E4
SSET	S^S	$E6
SWAIT	S^W	$E7
TEXT	T^E	$F1
TRANS	T^R	$DE

```
TUNE ............ T^U .... $E0
VOLUME= .......... V^O .... $E2
```

7.3.3 Speicherbelegungen

Im folgenden sei eine kurze Übersicht über die Speicherbereiche gegeben, die Supergraphik 64 im Computer belegt (gemeint ist stets RAM):

```
$0000-00FF: diverse Arbeitsspeicher
$0100-01FF: normaler Stack
$0200-03FF: unverändert genutzt
$0400-07E7: Text-LGR-Speicher
$0800-....: Basic-Speicherbereich
$7700-9FFF: Supergraphik - Betriebssystem
$A000-BFFF: Graphikseite 2
$C000-C3FF: Graphikseite 1-Farbe (Videoram)
            Spritevektoren
$C400-C5FF: Hardcopy-Routine
$C600-C7FF: diverse Speicher
$C800-CBFF: Graphikseite 2-Farbe (Videoram)
$CC00-CFFF: Graphikseite 2-Farbe (Farbram)
$D000-D1FF: Spritebewegungsregister
$D200-D5FF: Spritedefinitionen
$D600-DFFF: diverse Massen-Hilfsspeicher
            (f. RENUMBER/GMOVE/PAINT)
$E000-FFFF: Graphikseite 1

überlagert:
$D800-DBFF: originaler Farbram 1
```

Sie sehen, bei der Konzeption der Supergraphik mußte auf jedes Byte geachtet werden. Der zur Verfügung stehende Speicherplatz wurde 100 prozentig genutzt: Sage und schreibe 34K RAM werden von der Supergraphik zusätzlich genutzt. Trotzdem gehen dem Basic-Programmierer nur ganze 10K Speicherplatz verloren. Auch an anderer Stelle holt die Supergraphik 64 gerade noch das aus Ihrem Rechner, was möglich bzw. vertretbar ist - ich hoffe, zu Ihrem Nutzen!

7.4 CAD-Zeichner

7.4.1 Vollständiges Listing: CAD-Zeichner

Im folgenden finden Sie ein vollständiges Listing des im 2. Kapitel vorgestellten CAD-Zeichners. Wie Sie wissen, wurde er dort durch das gesamte Kapitel hindurch entwickelt und damit aus Gründen der Platzersparnis nie vollständig abgedruckt. Das wird hiermit nachgeholt:

```
100 REM *********************
110 REM **                 **
120 REM **  CAD-ZEICHNER   **
130 REM **                 **
140 REM *********************
150 REM
160 GMODE 0,1 : GCLEAR : SCOL= 0,0
170 X=160 : Y=100 :REM KOORDINATEN INIT
175 XA=320 : YA=200 :REM X- UND Y-AUFLOESUNG
180 POKE 650,255    :REM ALLE TASTEN AUF REPEAT
190 REM
200 REM EINGABESCHLEIFE:
210 REM
220 FL=0 :REM FLAG ZURUECKSETZEN
230 PLOT 2,X,Y : PLOT 2,X+1,Y :REM PUNKT INVERTIEREN
240 FL=ABS(FL-1) :REM FLAG 0<->1 WECHSELN
250 FOR Z=1 TO 30
260 GET A$ : IF A$="" THEN NEXT Z : GOTO 230
270 IF FL=1 THEN : PLOT 2,X,Y : PLOT 2,X+1,Y :REM PUNKT WIEDER
ZURUECKSETZEN
280 REM
290 REM
300 REM VERZWEIGUNG:
310 REM
320 A=ASC(A$)
330 IF A=17  THEN GOSUB 1300 :REM RUNTER
340 IF A=145 THEN GOSUB 1240 :REM HOCH
350 IF A=29  THEN GOSUB 1340 :REM RECHTS
360 IF A=157 THEN GOSUB 1390 :REM LINKS
370 IF A=133 THEN GOSUB 1010 :REM F1
380 IF A=137 THEN GOSUB 1040 :REM F2
390 IF A=134 THEN GOSUB 1070 :REM F3
400 IF A=138 THEN GOSUB 1100 :REM F4
410 IF A=135 THEN GOSUB 1130 :REM F5
420 IF A=139 THEN GOSUB 1160 :REM F6
430 IF A=136 THEN GOSUB 1190 :REM F7
440 IF A=140 THEN GOSUB 1220 :REM F8
450 IF A= 19 THEN CF=ABS(CF-1) :REM SCHNELL/LANGSAM
```

```
460 IF A$="Q" THEN:GMODE 0,0 : END
470 IF A$="L" THEN FZ=1 : GOSUB 1450 :REM LINIE
480 IF A$="R" THEN FZ=2 : GOSUB 1450 :REM RECHTECK
490 IF A$="B" THEN FZ=3 : GOSUB 1450 :REM BOX
500 IF A$="N" THEN GOSUB 5040        :REM NETZ ZEICHNEN
510 IF A$="K" THEN FZ=4 : GOSUB 1450 :REM KREIS
520 IF A$="T" THEN FZ=5 : T$="" : GOSUB 1450 :REM TEXT SCHREIBEN
900 GOTO 220
960 REM
970 REM
980 REM AUSFUEHRUNG:
990 REM
1000 REM F1: PUNKT SETZEN UND RECHTS
1010 PLOT ,X,Y : GOTO 1340
1020 REM
1030 REM F2: PUNKT LOESCHEN UND RECHTS
1040 PLOT 1,X,Y : GOTO 1340
1050 REM
1060 REM F3: PUNKT SETZEN UND LINKS
1070 PLOT ,X,Y : GOTO 1390
1080 REM
1090 REM F4: PUNKT LOESCHEN UND LINKS
1100 PLOT 1,X,Y : GOTO 1390
1110 REM
1120 REM F5: PUNKT SETZEN UND HOCH
1130 PLOT ,X,Y : GOTO 1240
1140 REM
1150 REM F6: PUNKT LOESCHEN UND HOCH
1160 PLOT 1,X,Y : GOTO 1240
1170 REM
1180 REM F7: PUNKT SETZEN UND RUNTER
1190 PLOT ,X,Y : GOTO 1300
1200 REM
1210 REM F8: PUNKT LOESCHEN UND RUNTER
1220 PLOT 1,X,Y : GOTO 1300
1230 REM
1240 REM CURSOR HOCH:
1250 IF CF=1 THEN Y=Y-9
1260 Y=Y-1 : IF Y<O THEN Y=YA+Y
1270 RETURN
1280 REM
1290 REM CURSOR RUNTER:
1300 IF CF=1 THEN Y=Y+9
1310 Y=Y+1 : IF Y>=YA THEN Y=Y-YA
1320 RETURN
1330 REM
1340 REM CURSOR RECHTS:
1350 IF CF=1 THEN X=X+9
1360 X=X+1 : IF X>=XA THEN X=X-XA
1370 RETURN
1380 REM
```

```
1390 REM CURSOR LINKS:
1400 IF CF=1 THEN X=X-9
1410 X=X-1 : IF X<0 THEN X=XA+X
1420 RETURN
1430 REM
1440 REM OBJEKTE ZEICHNEN:
1450 X1=X : Y1=Y :REM STARTKOORDINATEN
1455 IF FZ=4 THEN X=10 : Y=189 :REM KEIS
1460 X2=X : Y2=Y :REM ENDKOORDINATEN
1465 SL=2 : WA=0 : WE=359 :REM INITIALISIERUNG CIRCLE
1470 ZM=2 : FL=0 :REM FLAG ZURUECKSETZEN
1480 GOSUB 1730  :REM LINIE, FRAME, FILL,...
1490 FL=ABS(FL-1)
1500 FOR Z=1 TO 30
1510 GET A$ : IF A$="" THEN NEXT Z : GOTO 1480
1520 IF FL=1 THEN GOSUB 1730  :REM WIEDER ZURUEKSETZEN
1530 A=ASC(A$)
1540 IF A=17  THEN GOSUB 1300 :REM RUNTER
1550 IF A=145 THEN GOSUB 1240 :REM HOCH
1560 IF A=29  THEN GOSUB 1340 :REM RECHTS
1570 IF A=157 THEN GOSUB 1390 :REM LINKS
1580 IF A=133 THEN ZM=0 : GOTO 1740  :REM F1
1590 IF A=137 THEN ZM=1 : GOTO 1740  :REM F2
1600 IF A=134 THEN ZM=0 : GOSUB 1740 :REM F1
1610 IF A=138 THEN ZM=1 : GOSUB 1740 :REM F2
1620 IF A= 20 THEN X=X1 : Y=Y1 : RETURN :REM DEL => NICHT ZEICHNEN
1630 IF A= 19 THEN CF=ABS(CF-1)
1640 IF (A>=32 AND A<=127) OR A>=160 THEN IF LEN(T$)<=40 THEN T$=T$+A$
:REM TEXT
1690 REM
1700 X2=X : Y2=Y
1710 GOTO 1470
1720 REM
1730 ON FZ GOTO 1800,1900,1900,2050,2100 :REM SPRUNG ZU: LINIE, FRAME,
FILL, ...
1740 ON FZ GOTO 1800,1900,2000,2050,2100 :REM SPRUNG ZU: LINIE, FRAME,
FILL, ...
1770 REM
1780 REM
1790 REM LINIE ZEICHNEN/LOESCHEN:
1800 PLOT ZM,X1,Y1 TO X2,Y2
1810 RETURN
1870 REM
1880 REM
1890 REM RAHMEN ZEICHNEN/LOESCHEN:
1900 FRAME ZM,1,X1,Y1 TO X2,Y2
1910 RETURN
1970 REM
1980 REM
1990 REM BOX ZEICHNEN/LOESCHEN:
2000 FILL ZM,X1,Y1 TO X2,Y2
```

```
2010 RETURN
2020 REM
2030 REM
2040 REM KREIS ZEICHNEN/LOESCHEN:
2050 CIRCLE ZM,SL,X1,Y1,X2,199-Y2,WA,WE
2060 RETURN
2070 REM
2080 REM
2090 REM TEXT ZEICHNEN/LOESCHEN:
2100 IF T$="" THEN:PLOT ZM,X2,Y2 : GOTO 2120
2110 TEXT ZM,T$,X2,Y2,1
2120 RETURN
5000 REM
5010 REM
5020 REM NETZ BZW. RASTER ZEICHNEN
5030 REM
5040 GMODE ,8,161,250 :REM TEXTFENSTER OEFFNEN
5050 PRINT CHR$(147)  :REM TEXT LOESCHEN
5060 POS= 0,20
5070 PRINT "PUNKT-/LINIEN-NETZ (P/L)"
5080 PRINT "NETZPUNKTE-EINHEIT"
5090 POS= 26,20
5100 INPUT PF$ : IF PF$<>"L" AND PF$<>"P" THEN 5090
5110 POS= 26,21
5120 INPUT NE$ : IF VAL(NE$)<2 THEN 5110
5130 GMODE ,1  :REM GRAPHIK WIEDER VOLL
5140 NE=VAL(NE$)
5150 IF PF$="L" THEN 5220
5160 REM
5170 REM PUNKTNETZ:
5180 FOR XN=0 TO 319 STEP NE
5190 FOR YN=0 TO 199 STEP NE
5200 PLOT 2,XN,YN : PLOT 2,XN+1,YN
5210 NEXT YN
5220 NEXT XN
5230 RETURN
5240 REM
5250 REM PUNKTNETZ:
5260 FOR XN=0 TO 319 STEP NE
5270 PLOT 2,XN,0 TO XN,199
5280 NEXT XN
5290 FOR YN=0 TO 199 STEP NE
5300 PLOT 2,0,YN TO 319,YN
5310 NEXT YN
5320 RETURN
```

7.4.2 Kurzanleitung

Im Hauptmodus reagiert das Programm auf die folgenden Kommandos:

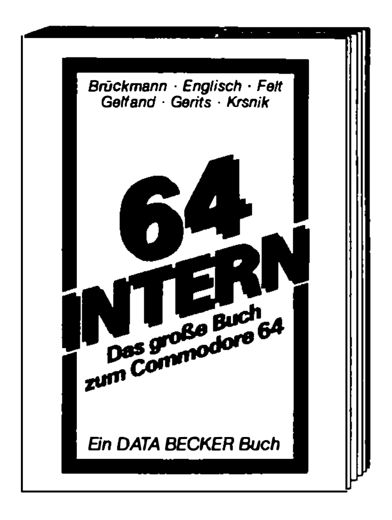
```
CRSR hoch   - bewegt blinkenden Punkt hoch
CRSR runter - bewegt blinkenden Punkt hinunter
CRSR rechts - bewegt blinkenden Punkt nach rechts
CRSR links  - bewegt blinkenden Punkt nach links
f1          - Punkt setzen und CRSR rechts
f2          - Punkt löschen und CRSR rechts
f3          - Punkt setzen und CRSR links
f4          - Punkt löschen und CRSR links
f5          - Punkt setzen und CRSR hoch
f6          - Punkt löschen und CRSR hoch
f7          - Punkt setzen und CRSR hinunter
f8          - Punkt löschen und CRSR hinunter
clr/home    - Wechsel zwischen schnell und langsam
L           - Linie zeichnen
R           - Rechteck zeichnen
B           - Ausgefülltes Rechteck (Box) zeichnen
K           - Kreis zeichnen
T           - Text schreiben
N           - Rasternetz in die Graphik einzeichnen
Q           - Beenden des Programmes
```

```
Figurenmodus (Linie, Rechteck, Box, Kreis, Text):

CRSR hoch   - Figurenendpunkt hoch
CRSR runter - Figurenendpunkt hinunter
CRSR rechts - Figurenendpunkt nach rechts
CRSR links  - Figurenendpunkt nach links
clr/home    - Cursorgeschwindigkeit
DEL         - Figurenmodus abbrechen
f1          - Figur zeichnen
f2          - Figur löschen
f3          - Figur zeichnen und mit gleichem
              Startpunkt die nächste Figur
f4          - Figur löschen und mit gleichem
              Startpunkt die nächste Figur
```

8. Stichwortverzeichnis

64 Intern ist ein Standardwerk zum Commodore 64, das vom ausführlich dokumentierten ROM-Listing über die detaillierte Hardwarebeschreibung bis zu nützlichen BASIC-Erweiterungen alles enthält, was man zum professionellen Einsatz des Commodore 64 wissen muß.



Aus dem Inhalt:

- Speicherbelegungspläne
- Der Soundcontroller und seine Programmierung
- Die Handhabung des AD-Wandlers
- Der Videocontroller
- Programmierung von Farbe und Grafik
- Die Zeichengenerator-Schnittstelle
- Sprites
- Ein-/Ausgabesteuerung
- Timer und Echtzeituhr
- Joystickprogrammierung
- So arbeitet der BASIC-Interpreter
- Mathematische Routinen – selbst entwickelt
- Der serielle IEC-Bus
- Programmierung der RS-232
- Die Belegung der Zero-Page
- Der Commodore-64-Schaltplan

Angerhausen, Brückmann, Englisch, Gerits
64 Intern
Hardcover, 628 Seiten, DM 69,–
ISBN 3-89011-000-2

64 Tips & Tricks Band 1, das auflagenstärkste deutsche Computerbuch, bringt in einer komplett überarbeiteten und stark erweiterten Neuauflage alle Tips & Tricks auf einen Blick. Sparen Sie sich das lästige Blättern und Suchen in anderen Büchern und Zeitschriften – mit dem Original können Sie Ihre Zeit sofort zum Programmieren verwenden! Jetzt noch mehr Fakten: BASIC-Programmierung effektiver und besser, Grafik für Fortgeschrittene, Soundprogrammierung, die Benutzerschnittstellen, die Peripherie, Befehlserweiterungen, Schnittstellen und ein ganzes Kapitel mit Kurztips.



Aus dem Inhalt:

- Die drei Uhren des C-64
- Errechnen beliebiger Formeln
- Bubblesort in Assembler
- Grafikanimation
- Hochauflösende 3-D-Grafik
- Synthesizer in Stereo
- Maussimulation mit dem Joystick
- GEM-Simulation auf dem C-64
- Kopierschutz
- User-Port
- Datenübertragung zu anderen Rechnern
- Directory ohne Programmverlust

Englisch, Gerits, Hartwig, Löffelmann, Thrun
64 Tips & Tricks Band 1
418 Seiten, DM 49,–
ISBN 3-89011-001-0

Bücher zum Commodore 64

64 Tips & Tricks Bd. 2 enthält eine Fülle hochkarätiger Programme, Anregungen und viele nützliche Routinen. Ein Buch, das für jeden, der auf dem COMMODORE 64 eigene Programme schreiben will, eine unentbehrliche Hilfe ist.



Aus dem Inhalt:

- Softwareschutz
- Befehlserweiterung – selbst gemacht
- Grafik – Zeichendefinition
- Spieleprogrammierung
- Betriebssystem: ROM in RAM
- Betriebssystem-Routinen
- Wie speichert der Computer eine BASIC-Zeile?
- Hardware-Tips
- Laufschrift
- Arbeiten mit zwei Bildschirmen
- Modifiziertes Input

und vieles mehr ...

Weltner, Homig, Trapp
64 Tips & Tricks Band 2
259 Seiten, DM 39,–
ISBN 3-89011-065-7

Dieses Buch erklärt Ihnen leichtverständlich den Umgang mit Peeks & Pokes. Es enthält eine Beschreibung aller nutzbaren Speicheradressen und führt in die Hardware Ihres C64 ein. Die vielen sofort einsetzbaren Programme haben schnell dafür gesorgt, daß dieses Buch schon in der ersten Auflage zu einem unverzichtbaren Nachschlagewerk für jeden interessierten Programmierer wurde.



Aus dem Inhalt:

- Die Arbeitsweise der CPU
- Was ist ein Betriebssystem
- Wie arbeitet der BASIC-Interpreter
- Beschreibung und Nutzung der Zeropage
- Pointer & Stacks
- Speicherbelegungsplan
- Massenspeicherung & Peripherie
- Die Spriteregister
- Interruptprogrammierung
- Leichtverständliche Einführung in die Maschinensprache

Liesert
Peeks & Pokes zum C 64
196 Seiten, DM 29,–
ISBN 3-89011-032-0

DAS STEHT DRIN:

Ein Grafikbetriebssystem für Ihren C64 – das bietet Ihnen dieses Buch. Und natürlich die Möglichkeit, wirklich alles über die C64-Grafik zu lernen. Der Schwierigkeitsgrad ist dabei frei wählbar: vom Einsteigen mit Hunderten von neuen Befehlen bis zum Programmieren von Grafikalgorithmen in Assembler. Vom CIRCLE-Befehl bis zu seinem kommentierten Source-Code.

- Multicolorgrafik
- CAD-Zeichner – das Zeichenpaket
- Grafikfenster durch Raster-Interrupt
- Text in hochauflösender Grafik
- Shapes auf dem C64
- 136 Farben auf dem C64
- Einladen von Fremdgrafiken
- Viele Utilities
- 3-D-Grafik
- Animation und Scrolling
- Text und Grafik auf dem Low-Res-Bildschirm
- Kommentiertes Source-Listing von Supergraphik

UND GESCHRIEBEN HAT DIESES BUCH:

Wer weiß wirklich alle Tips und Kniffe eines Top-Programms? Wer kennt die verborgenen Möglichkeiten? Wer kann ein vollständiges Paket wie Supergraphik am besten dokumentieren und seinen Aufbau erklären? Natürlich der Programmierer selbst: Axel Plenge. Er hat aber nicht nur das optimale Wissen, wenn es um seine Befehlserweiterung Supergraphik geht, sondern auch mit mehreren Büchern bewiesen, daß er anschaulich und interessant schreiben kann.

